VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

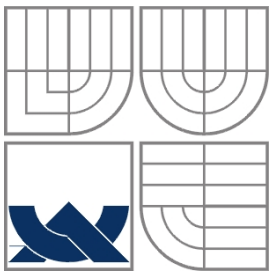
VESTAVĚNÝ TERMINÁL PRO PODPORU ORGANIZACE ZÁVODŮ V ORIENTAČNÍCH SPORTECH

BAKALÁŘSKÁ PRÁCE
BACHELOR´S THESIS

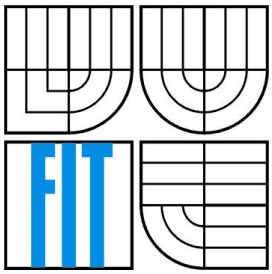
AUTOR PRÁCE
AUTHOR

TOMÁŠ KAVAN

BRNO 2013



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

TERMINÁL PRO ORGANIZACI ZÁVODŮ V ORIENTAČNÍCH SPORTECH

EMBEDDED SYSTEM FOR ORIENTEERS RACE ORGANIZATION SUPPORT

BAKALÁŘSKÁ PRÁCE
BACHELOR´S THESIS

AUTOR PRÁCE
AUTHOR

TOMÁŠ KAVAN

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. VÍTĚZSLAV BERAN, Ph.D.

BRNO 2013

Abstrakt

Práce pojednává o návrhu a výrobě prototypu terminálu pro podporu organizace závodů v orientačních sportech. Cílem práce je integrace všech důležitých komponent výpočetní techniky používaných při organizaci závodu nebo tréninků v orientačních sportech do podoby přenosného kompaktního vestavěného systému o velikosti běžného terminálu pro akceptaci platebních karet. Sekundárním cílem je tvorba multiplatformní Open Source knihovny pro komunikaci s jednotkami SportIdent.

Klíčová slova

Orientační sporty, RFID, časomíra, vestavěné systémy, sériová komunikace, iOS, SportIdent, Arduino

Abstract

This thesis deals with the design and production of prototype terminal for supporting organization within orienteering races. The aim of this work is to integrate all important computer facilities, currently used in the practice of orienteering races or training, into a portable compact embedded system in the size of a common terminal for accepting credit cards. The secondary objective of this thesis is creation of cross-platform Open Source library for communication with SPORTIdent control units.

Keywords

Orietneering, RFID, timekeeping, embedded systems, serial communication, iOS, SportIdent, Arduino

Citace

Kavan Tomáš: Vestavěný terminál pro podporu organizace závodů v orientačních sportech. Brno, 2013, bakalářská práce, FIT VUT v Brně.

Vestavěný terminál pro podporu organizace závodů v orientačních sportech

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením Ing. Vítězslava Berana, Ph.D. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Tomáš Kavan
5. února 2013

Poděkování

Chtěl bych poděkovat firmě SPORTIdent za poskytnutí kontrolní jednotky systému SPORTIdent, dále všem svým respondentům, oddílu orientačního běhu TBM za podporu při testování terminálu a především svému vedoucímu práce za nehynoucí trpělivost, podporu a výborné nápady při tvorbě této práce. Mé poděkování také patří mému kolegovi Ing. Václavu Blažkovi, který mi pomohl zapůjčením nářadí a svými zkušenostmi při přípravě obalu terminálu.

Obsah

Úvod

Orientační sporty jsou velké, ale méně známé sportovní odvětví. V porovnání s jinými sporty jsou ty orientační velmi specifické z pohledu používaných výpočetních systémů a technologií. Dnes se výpočetní technologie používají pro široké spektrum činností v OSP[1]. Jedná se především o navigační, triangulační a kreslící systémy používané při mapování, speciální časoměrné systémy v kombinaci s bezdrátovými technologiemi RFID pro organizaci závodních i tréninkových událostí a databázové a webové aplikace pro podporu administrativních činností jako jsou například přihlašovací, platební a výsledkový servis.

V této práci se zaměřím na návrh a vývoj zařízení pro podporu organizace tréninkových a závodních událostí. Mým prvořadým cílem – vyplývajícím z rozsáhlého šetření mezi uživateli – je navrhnout a vytvořit přenosné integrované terminálové řešení obsahující všechny důležité prvky výpočetní techniky i softwarového vybavení (dále jen terminál). Terminál by měl výrazným způsobem zjednodušit zpracování přihlášek, startovní listiny a výsledků. Zároveň by měl být pro organizaci závodů jediným nezbytným kusem výpočetní techniky. Mým druhořadým cílem je, v rámci vývoje prototypu, připravit a zveřejnit multiplatformní Open Source knihovnu pro komunikaci s dominantním časoměrným systémem SPORTIdent.

Práce je rozdělena do třech částí. V první kapitole nastiňuji teoretický základ potřebný pro detailní orientaci v dalším textu. V krátkosti proberu problematiku pravidel OSP především s důrazem na cíl práce. Dále se pak věnuji současným řešením používaným při organizaci závodních a tréninkových událostí v OSP. Jedná se především o časoměrné systémy v OSP (kapitola 1.3), standardy a formáty pro výměnu dat v OSP (kapitola 1.4) a softwarové nástroje (kapitola 1.5). Dále se pak věnuji teorii potřebné pro návrh a tvorbu samotného terminálu.

V druhé kapitole popisuji návrh samotného terminálu. V úvodu shrnuji výsledky šetření mezi uživateli (kapitola 2.1). Po té už probírám návrh jednotlivých komponent terminálu, SPORTIdent knihovny a iOS aplikace EasyEvent určené pro obsluhu terminálu.

Ve třetí kapitole jsem shrnul poznatky získané během vývoje prototypu a jeho reálného zkušebního nasazení v oblastním závodě v orientačním běhu.

[1] Orientační sporty
[2] Operační systém pro mobilní telefony od firmy Apple
[3] http://www.orientacnisporty.cz/cz/co-jsou-orientacni-sporty/

1 Teoretický základ

Pro pochopení smyslu a konstrukce terminálu si je třeba osvojit teoretický základ. Tomu věnuji tuto kapitolu. U každé části se snažím uvést dostatečné množství referenční literatury, tak aby si čtenář mohl o problematice vytvořit hluboký rozhled. Zároveň se snažím teoretický základ shrnout tak, aby další text bylo možné číst bez studia dalších zdrojů.

Studiem první části (kapitoly 1.2, 1.3, 1.4 a 1.5) by měl čtenář získat přehled o tom, co to orientační sporty jsou a jaká jsou současná řešení z oblasti výpočetní techniky používaných v OSP. Důraz kladu výlučně na část podpory organizace tréninkových a závodních událostí.

V druhé části (kapitoly 1.6 a 1.7) se potom čtenář dozví o technologiích relevantních pro návrh a konstrukci terminálu.

V práci předpokládám některé čtenářovy prerekvizitní znalosti. Většinou se jedná o elementární znalosti z oblasti informatiky a materiály pro jejich studium jsou volně a hojně dostupné.

Pro správné pochopení práce je třeba orientace v základech programování a jazyce C/C++. Dále předpokládám orientaci v abstraktních datových typech a datových strukturách. Přepokládám znalost značkovacího jazyka XML a orientaci v problematice mikrokontrolérů a vestavěných systémů [1]. Pro pochopení aplikace EasyEvent pak znalost vývoje aplikací pro zařízení s operačním systémem iOS[2], vývojového prostředí Xcode, iOS SDK a frameworku Core Data [2].

1.1 Pravidla orientačních sportů

Orientační sporty jsou moderní sportovní odvětví vytrvalostního charakteru, při němž je nutno se správně a rychle orientovat v neznámém terénu za pomocí podrobné mapy a buzoly. Při závodě v OSP závodník absolvuje předem určenou trať vedoucí mezi kontrolními stanovišti v nejkratším možném čase[3] [3].

Orientační sportovci jsou sdruženi do klubů a dále pak do národních organizací. Existuje celosvětová organizace orientačních sportovců – International Orienteering Federaion (IOF) – určující mezinárodní pravidla a standardy v OSP a pořádající nadnárodní mistrovství[4]. Česká národní organizace – Český svaz orientačních sportů (ČSOS) – upravuje pravidla a standardy pro české soutěže[5].

V orientačních sportech existuje několik disciplín. Jedná se o orientační běh, orientační závody na horských kolech, lyžařský orientační běh a trail-O. Z pohledu časoměrných systémů a organizace závodních a tréninkových událostí nejsou mezi těmito disciplínami žádné rozdíly.

Veškeré potřebné dokumenty s pravidly OSP jsou k naleznutí v [4] a [5].

1.1.1 Závod v OSP

Závod v OSP je většinou rozdělen na více kategorií (typicky 10 - 20) podle pohlaví a věku. Závodníci jedné kategorie v rámci závodu absolvují stejnou trať. Nestartují však dohromady, ale v časových dostupech závislých na délce tratě. Různé kategorie mají v závodě různé tratě. Každá trať má několik kontrolních stanovišť, které je závodník povinen absolvovat v předem určeném pořadí (pořadí je zakresleno v mapě). Kontrolní stanoviště může být společné pro tratě různých kategorií. Pořadí závodníků v kategorii určuje čas potřebný k absolvování celé tratě. Měření času začíná ve

[2] Operační systém pro mobilní telefony od firmy Apple
[3] http://www.orientacnisporty.cz/cz/co-jsou-orientacni-sporty/
[4] http://orienteering.org
[5] http://www.orientacnisporty.cz

startovní čas závodníka a končí překonáním cílové pásky závodníkem. Vzhledem k intervalovému startu nemusí být – a většinou nebývá – vítězem první závodník v cíli.

Existují speciální případy tratí s různými specifiky. Časoměrné systémy musí tato specifika zohlednit a integrovat. Jedná se především o níže uvedené typy.

Štafetový závod a závod družstev

Závodníci se sdruží do 3 – 6 členných družstev a vyrážejí na trať postupně. Tedy předáním pomyslného štafetového kolíku. Celkový čas štafety/družstva se počítá součtem časů všech členů.

Trať je většinou postavena okruhy (viz dále) a každý člen štafety absolvuje jeden okruh.

Závod s volným pořadím kontrolních stanovišť

Kontrolní stanoviště na trati nemají určeno pevné pořadí jejich absolvování. Závodník pořadí volí sám po startu. Cílem je absolvovat všechna kontrolní stanoviště v nejkratším možném čase.

Závod s volným pořadím ohodnocených kontrolních stanovišť

Speciální případ závodu s volným pořadím kontrolních stanovišť. Jednotlivé kontroly mají přiřazeno bodové ohodnocení. Závodník nemusí absolvovat všechna stanoviště. Na závod má předem daný časový limit. Pořadí závodníků určuje jejich bodový zisk z kontrolních stanovišť ponížený o případnou penalizaci za překročení času.

Závod s hromadným startem

Závodníci nestartují v časových odstupech za sebou, ale všichni zároveň. Většinou bývá tento typ kombinován s tratí s okruhy.

Více-etapový závod s handicapovým startem poslední etapy

Pořadí závodníka určuje součet časů z jednotlivých etap závodu. Startovní čas závodníka v poslední etapě je roven jeho odstupu na prvního závodníka kategorie. Pořadí závodníků v cíli poslední etapy je tedy celkovým pořadím. Používá se především na vícedenních závodech.

Závod s okruhy

Trať jedné kategorie prochází vícekrát jedním kontrolním stanovištěm, tvoří tedy okruhy. Každý závodník absolvuje stejnou trať, jen každý okruh v jiném pořadí.

1.2 Časoměrné systémy v orientačních sportech

Časoměrný systém při pořádání především závodních, ale i tréninkových událostí v OSP hraje zásadní roli v jejich hladkém a férovém průběhu. Časoměrný systém neslouží pouze pro měření závodníkova času stráveného na trati, ale kontroluje i správnost pořadí označování průchodu kontrolními stanovišti (ražení/punching).

Dříve se pro ražení používal mechanický systém ražení hrotů speciálními kleštěmi do papírového kontrolního průkazu závodníka. Každé kleště v závodě byli unikátní posazením hrotů. Návštěvu kontrolních stanovišť v určeném pořadí nebylo možné kontrolovat jinak než přímou kontrolou kontrolního průkazu na trati. Měření času v cíli potom vyžadovalo precizní synchronizaci s odebíráním kontrolních průkazů tak, aby každému závodníkovi byl přiřazen správný cílový čas.

S příchodem technologie RFID se v OSP začaly objevovat systémy spojující měření času a ražení. Papírový kontrolní průkaz nahradil elektronický čip a kleště na kontrolních stanovištích

elektronické kontrolní jednotky. Vše s technologií RFID. Výhoda výrazného zjednodušení zpracování závodních dat jasně převážila nad nevýhodou vyšší pořizovací ceny systému.

1.2.1 Systém SPORTIdent

Dnes se celosvětově v OSP používá systém elektronického ražení SportIdent[6], který se tak stal standardem. Většina závodníků vlastní elektronický průkaz (SI čip). Na závodech si je možné průkaz také za poplatek zapůjčit.

Architektura systému

Systém SportIdent (SI) používá technologii bezkontaktního čipu RFID. Systém se skládá z elektronických průkazů (SI čipů), kontrolních jednotek a základových stanic.

Každý SI čip obsahuje RFID čip s interní anténou a má své unikátní číslo (Obrázek 1.1). Existuje více verzí SI čipů lišící se rychlostí zápisu, počtem pozic pro ražení, tvarem a možností umístění dodatečných informací. Kromě běžných razících pozic obsahuje každý SI čip pozice pro záznam času svého vymazání, kontroly vymazání, průchodu startem a průchodu cílem.

Kontrolní jednotka je vestavěný systém s vlastní baterií, mikrokontrolérem a anténou pro bezkontaktní komunikaci. Kontrolní jednotka zapisuje data na SI čipy automaticky po jejich vložení do otvoru v těle jednotky (Obrázek 1.1). Jednotka do každého čipu zapíše na první volnou pozici své unikátní identifikační číslo, kód kontroly a čas průchodu. Jednotka se programuje pomocí bezdotykové komunikace se základovou stanicí. Do komunikace mezi SI čipem a kontrolní jednotkou není možné zasahovat.

Kontrolní jednotka běžně funguje v operačním režimu „kontrola“. Pomocí základové stanice může být přepnuta do operačních režimů „mazání“, „kontrola vymazání“, „start“ a „cíl“. V těchto operačních režimech kontrola zapisuje čas průchodu do speciálních razících pozic SI Čipu.

Základová stanice je klasická kontrolní jednotka, která navíc obsahuje sériové rozhraní pro komunikaci s externím počítačovým systémem a softwarem. Stanice je schopna programovat čas, kód a operační režim kontrolních jednotek, stejně jako plánovat jejich automatické zapnutí a vypnutí. Pro komunikaci jednotek se základovou stanicí se používá jejich bezkontaktní propojení speciálním magnetem válcového tvaru. Stanice je také schopna přečíst kompletní obsah vloženého SI čipu a odeslat jej sériovou linkou externímu počítačovému systému. Veškerá činnost základové stanice je řízena po sériové lince.

Základové stanice je dodávána s rozhraním RS-232 nebo USB. Stanice s USB rozhraním obsahuje čip vyvíjený firmou Future Techonolgy Devices Ltd. (FTDI) [6]. Firma FTDI dodává multiplatformní ovladač pro USB sériový port[7]. Kontrolní jednotku ve verzi RS-232 i USB je možné připojit k široké škále počítačových systémů.

Firma SportIdent ke svému systému poskytuje knihovnu pro komunikaci s operačním systémem Microsoft Windows. A pro stejný systém poskytuje i sadu základních nástrojů pro práci se základovými stanicemi. Jiné operační systémy firma SportIdent nepodporuje. Implementaci komplexnějších systémů pro podporu organizace tréninkových a závodní OSP událostí ponechává na třetích stranách. Blíže se těmto systémům budu věnovat v kapitole 1.5.2.

[6] http://www.sportident.com
[7] http://www.ftdichip.com/FTDrivers.htm

Obrázek 1.1 – SI čip verze 6 s unikátním číslem a SI jednotka verze 8 – klasická kontrola (zdroj: http://www.sportident.com)

Komunikační protokol jednotky

Firma SportIdent vydává a spravuje dokumentaci komunikačního protokolu základové stanice s externím počítačovým systémem po sériové lince. Podrobná dokumentace je po registraci dostupná na webu výrobce systému [7]. Komunikační protokol dovoluje řídit činnost základové stanice. Především se jedná o inicializaci komunikace, čtení dat z vloženého SI čipu, programování přiložených kontrolních jednotek, přístup k záložní paměti základové stanice a přiložených kontrolních jednotek.

Existuje několik verzí základových a kontrolních jednotek i samotných SI čipů. Existuje více verzí komunikačního protokolu. Níže uvádím tabulku kompatibility jednotlivých verzí kontrolních a základních jednotek s verzí protokolu (tabulka 1.1). Kontrolní jednotky jsou zpětně kompatibilní. Vzhledem k ukončení výroby starších jednotek se budu dále věnovat jen protokolu verze 5.

Typ jednotky	**Verze software jednotky**	**Verze protokolu**
BSF3-x, BSM3-x, BSx4, BSx6	4.06	4.0x
BSx7, BSx8	5.29	5

Tabulka 1.1 – kompatibilita základových a kontrolních jednotek SI

Komunikační rozhraní je asynchronní a realizované nad standardní sériovou linkou RS-232. Spojení je konfigurováno jako poloviční duplex, bez paritních bitů, s jedním stop bitem a bez hardwarového handshake. Pro zahájení vysílání příkazu a jeho ukončení se používají počáteční (STX) a konečné (ETX) značky.

Každý příkaz je označen kódem. Kód příkazu má délku 1 byte. Příkazy s kódem menším než 0x80 jsou z protokolu verze 4. Příkazy s kódem 0x80 a větším jsou z protokolu verze 5. Společný formát zprávy definuje tabulka 1.2.

STX (0x02)	Kód příkazu	Tělo zprávy – proměnlivá délka	ETX (0x03)

Tabulka 1.2 – Obecný formát protokolu SI

Ve verzi protokolu 4 se řídící znaky (0x00 – 0x1F) v těle zprávy prefixovaly znakem oddělovače (DLE – 0x10). V protokolu verze 5 za kódem příkazu následuje byte s délkou těla zprávy a za zprávou 2 byte s kontrolním součtem následovaným ETX. V protokolu verze 5 se oddělovač (DLE) nepoužívá. Výpočet kontrolního součtu zprávy popisuje příloha 1.

Základová stanice může pracovat ve dvou komunikačních módech. V módu „auto send out“ automaticky přečte data z vloženého čipu a zašle je po sériové lince. V módu „handshake“ stanice oznámí vložení čipu a vyčkává na příkazy. Po vyjmutí čipu stanice opět oznámí událost.

Detailní popis instrukční sady, struktury záložní paměti a paměti SI čipu jsou uvedeny v dokumentaci dodavatele [7].

1.2.2 Systém Emit

Převážně v severských zemích se používá systém elektronického ražení Emit[8]. Pro tuto práci však není relevantní.

1.3 Data a standardy jejich výměny

Při zamyšlení nad tréninkovými a závodními událostmi v OSP z pohledu datového modelování je možné data rozdělit do jasně definovaných struktur, které jsou společné všem událostem. Dále je možné specifikovat toky datových struktur mezi jednotlivými systémy. S těmito poznatky je možné jednoduše vymodelovat komunikační protokol.

1.3.1 IOF Interface standard

Datovým modelováním se v OSP dlouhodobě zabývá expertní skupina v rámci organizace IOF. Tato skupina vydává a spravuje je „IOF Interface Standard“. V současné době se běžně používá verze 2.0.3 [8], která je však postupně nahrazována verzí 3.0 [9]. Standart je specifikací XML datového formátu pro výměnu OSP relevantních datových struktur mezi aplikacemi, databázemi a webovými službami.

IOF Interface standard definuje formát pouze pro výměnu dat. Pro analýzu a datové modelování systémů pro zpracování přihlášek a výsledků žádný standard neexistuje.

IOF Interface Standard verze 2.0.3

Standard je definovaný pomocí DTD[9] a definuje seznamy pro výměnu dat mezi systémy.

- **Seznam osob** umožňující zaslat seznam osob bez informací relevantních pro OSP. Jedná se tedy o adresář. Seznam osob se téměř pro komunikaci mezi systémy nepoužívá.
- **Seznam závodníků** umožňující zaslat celkový seznam závodníků. Používá se pro zasílání kompletní – například národní – databáze závodníků. Nepoužívá se jako seznam přihlášených závodníků k nějaké konkrétní události.
- **Seznam výsledků v žebříčku**
- **Seznam klubů** umožňující zaslat databázi klubů. Například národní nebo vztaženou ke konkrétní události.
- **Seznam událostí** umožňuje zaslat seznam naplánovaných událostí. Událostí se rozumí jeden jedno etapový nebo více etapový závod či trénink.
- **Seznam žádostí o dodatečné služby**. Dodatečná služba je poskytována během události a může se jednat například o ubytování či stravování. Poskytované dodatečné služby definuje událost.
- **Seznam přihlášek** umožňuje zaslat seznam závodníků přihlášených na konkrétní událost do konkrétní kategorie.

[8] Systém elektronické časomíry a záznamu ražení od Norského výrobce.
[9] Document type definition, česky definice typu dokumentu.

- **Startovní listina** poskytuje seznam startovních časů závodníků k jednotlivým etapám v události podle kategorií.
- **Výsledková listina** poskytuje seznam výsledů závodníků k jednotlivým etapám v události.
- **Data kategorií** umožňuje zaslat seznam kategorií v národní soutěži či ke konkrétní události.
- **Data běhů** umožňují zaslat kompletní nastavovací data události jako je seznam map, kontrol, atd.

Standard dále definuje datové typy společné všem seznamům. Definice jednotlivých typů přesahuje možnosti této kapitoly.

IOF Interface Standard verze 3.0

Standard je definovaný pomocí XML Schema[10]. Standard rozšiřuje předchozí verzi 2.0.3 a doplňuje ji o chybějící vlastnosti. Používá standardní formáty pro zápis data ISO8601. Přidává chybějící pole do adresáře osob a organizací. Zavádí podporu pro geolokaci a sledování závodníků a především širší podporu pro různé typy závodních událostí.

1.3.2 ČSOS Standard

Dalším z u nás používaných standardů je textový formát „ČSOS formát“ sloužící pro zasílání přihlášek a výsledkových listin závodů v unifikovaném textovém souboru. Dnes je spíše na ústupu.

1.4 Nástroje pro organizaci závodů

V kapitole nejdříve proberu postup při organizaci události v OSP, abych následně mohl rozebrat v současnosti dostupné nástroje (systémy) pro podporu přípravy události. Čtenář se dozví, že přípravu lze rozdělit do několika částí a že pro každou část existuje specializovaný software pro její podporu.

1.4.1 Příprava a organizace události v OSP

I když se samotná událost většinou odehrává v rámci jednoho či několika málo dnů, prvotní úvahy a přípravné práce začínají několik měsíců dopředu. Délka přípravných prací závisí na typu a rozsahu události. Velké mezinárodní soutěže jako je mistrovství světa se začíná plánovat až 2 roky předem, vícedenní závody přibližně rok dopředu a běžné oblastní jednodenní závody dva až tři měsíce před konáním akce.

Přípravu a organizaci události v OSP je možné rozdělit na různé části podle různých kritérií, pro tuto práci jsem zvolil dělení podle toku dat. Protože neexistuje žádný oficiální metodický pokyn či pravidla pro organizaci události v OSP, je uvedené dělení pouze ilustrativního charakteru a mělo by usnadnit orientaci v toku dat během organizace události. Obrázek 1.3 ilustruje činnosti (obdélníky) a toky dat mezi nimi (šipky s popisky). Pro toky dat zapsané tučným písmem je možné používat výměnu dat službami podle standardu IOF Interface Standard (viz. kapitola 1.4.1).

[10] Jazyk popisující strukturu XML dokumentu.

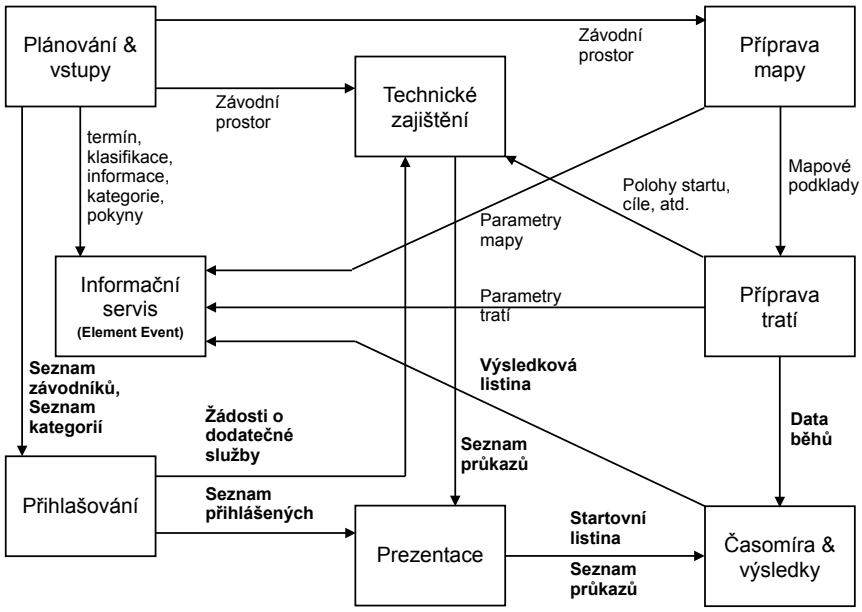


Schéma 1.1 – Tok dat během přípravy události v OSP

První fází přípravy události v OSP je **plánování a definice vstupů**. Jedná se především o definici termínu a místa konání události a sestavení realizačního týmu. Zároveň je v této fázi nutné definovat vstupy události. Tedy požadavky nadřazené organizace, pravidel a prováděcích pokynu, seznamu dostupných závodníků a požadovaných kategorií.

Dále je na celou událost možné pohlížet jako na projekt a při přípravě postupovat podle některé z metodik řízení projektů.

S **technickým zajištěním** události jsou spojeny administrativní a technické úkoly nutné pro uspořádání události. Většinou se jedná o vyřízení veškerých potřebných povolení v místě konání události, zapůjčení technických prostředků (ozvučení, časomíra, atd.), materiální zajištění události, zajištění dodatečných služeb pro závodníky jako je ubytování, stravování a příprava logistického plánu. Pro tuto práci je relevantní především seznam kontrolních průkazů k zapůjčení. Tento seznam dodává poskytovatel časomíry spolu se sadou čipů.

Informační servis je většinou realizován formou webových stránek a slouží k informování závodníků a zájemců o události. Vstupem jsou organizační informace (viz obrázek 1.3). Dále startovní a výsledková listina. Výstupem může, kromě webových stránek, být element Event připravený k přenosu do jiného systému (kalendáře, atd.) pomocí seznamu událostí s jediným elementem.

Přihlašování na závody je většinou realizováno pomocí univerzálního přihlašovacího systému. Vstupem do systému je národní či mezinárodní seznam závodníků a element Event s definicí kategorií, termínů, poplatků a dodatečných služeb. Výstupem je seznam přihlášek a seznam žádostí o dodatečné služby.

Přípravou mapy se rozumí mapování závodního prostoru události či revize existující mapy. Jedná se většinou o časově nejnáročnější akci přípravy události. Vstupem je definice závodního prostoru a podklady pro mapování. Pro mapování v terénu a kreslení mapy se používá specializovaný kreslicí a kartografický software. Výstupem jsou mapové podklady v tisknutelném formátu, který podporují systémy pro přípravu tratí.

Během **přípravy tratí** se provádí plánování a zákres tratí na mapové podklady vytvořené během přípravy mapy. V rámci přípravy tratí se dále přiřazují kategorie vytvořeným okruhům. Výstupem jsou data běhů pro událost a pozice a počet startovních, cílových a občerstvovacích stanovišť.

Prezentace je organizátorské stanoviště v místě události na které se všichni závodníci obracejí s administrativními požadavky jako je změna startovního času, pozdní dohlášky, platby, výpůjčky kontrolních průkazů, atd. V rámci prezentace se generuje a aktualizuje startovní listina. Vstupem pro prezentaci je seznam přihlášených závodníků a seznam průkazů určených k zapůjčení. Výstupem jsou podklady pro časomíru, tedy prezenční a startovní listina včetně čísel kontrolních průkazů jednotlivých závodníků.

V samotném průběhu události jsou **časomírou** sbírána data ze startu, cíle či radiových kontrol, vyhodnocován průběh závodu jednotlivých závodníků a poskytovány průběžné a na závěr celkové výsledkové listiny.

V dalších podkapitolách se budu zabývat v současné době dostupnými systémy pro podporu přípravy událostí v OSP. Vzhledem k cíli této práce budu detailněji popisovat jen systémy pro prezentaci a zpracování výsledků. Okrajově se potom zmíním o systémech pro přihlašování, pro podporu tvorby map a pro přípravu závodních dat.

1.4.2 Systémy pro prezentaci a zpracování výsledků

Systémy pro prezentaci a zpracování výsledků umožňují připojit výstupy časomíry a jejích prvků pro správu elektronických průkazů. Častou vlastností těchto systémů je specifický výstup pro divácké tabule a komentátory. V současné době existuje velké množství těchto systémů[11].

V současné době neexistuje řešení pro jiný operační systém než Microsoft Windows.

Orienteering Organiser

Celosvětově velmi oblíbený systém vyvíjený českým developerem jako hobby projekt[12]. Pouze pro operační systém Windows. Systém poskytuje velmi rozsáhlou škálu funkcí od přípravy závodních dat (viz. kapitola 1.5.4), přes administraci události, prezentaci, podporu časomíry, diváckých a komentátorských výstupů až po online výsledkový informační servis a analýzu postupů.

Systém podporuje import a export dat v kterékoliv fázi přípravy a průběhu události do širokého spektra formátů včetně standardů IOF Interface Standard verze 2.0.3 a ČSOB.

Systém je ovšem pro uživatele velmi neintuitivní a z pohledu návrhu uživatelského rozhraní velmi zaostává za moderními trendy. Uživatelské rozhraní je zastaralé a těžkopádné.

OB 2000

Český systém od tvůrce přihlašovacího systému OB Haná stále používaný převážně v moravských oblastech[13]. Podporuje tvorbu závodních dat bez grafické nadstavby. Pouze pro operační systém MS Windows 95 - 7.

Systém je pro uživatele velmi neintuitivní a z pohledu návrhu uživatelského rozhraní velmi zaostává za moderními trendy. Uživatelské rozhraní je zastaralé a těžkopádné. Samotná instalace systému je kvůli nutnosti upravovat systémové konfigurační soubory náročná.

OEvent

Zahraniční komerční software se zdařilým uživatelským rozhraním a intuitivním ovládáním[14]. Pouze pro platformu Microsoft Windows. Obsahuje podporu přípravy závodních dat a podporu systému SportIdent. Podporuje export a import z přihlašovacího systému Orienteering Online.

[11] http://o-wiki.net/index.php?title=Software
[12] http://www.orienteeringorganiser.com
[13] http://obhana.cz/sw/sw.htm
[14] http://www.oevent.org

4mila

Zahraniční volně šířený multiuživatelský síťový systém pro přípravu závodních dat, prezentaci a zpracování výsledků[15]. Implementovaný v jazyce JAVA, spustitelný pouze na platformě Microsoft Windows. Stanice používají tenkého klienta, serverová část staví na komunikaci přes HTTP protokol a databázi MySQL. Zdařilé uživatelské rozhraní a intuitivní ovládání. Podporuje široké spektrum importních a exportních formátů včetně IOF Interface standard 2.0.3 a 3.0.

SIME

SportIdent Mini Event[16]. Estonský minimalistický systém pro zpracování výsledků. Nepodporuje prezentaci. Z načtených závodních dat a vyčtených kontrolních průkazů generuje výsledkovou listinu. Výrobce časomíry SportIdent odkazuje SIME jako referenční software pro pořádání malých událostí v OSP.

1.4.3 Přihlašovací systémy

Většina přihlašovacích systémů je funkčně shodná, ale v každé geografické oblasti se většinou prosadil systém jiný. Jako zástupce této kategorie uvádím dva české a jeden mezinárodní přihlašovací systém.

ORIS

Oficiální přihláškový a výsledkový servis ČSOS. Webová aplikace[17]. Poskytuje vždy nejaktuálnější seznam závodníků a pořadatelům umožňuje export přihlášek ve formátech ČSOS a IOF Interface Standard verze 2.0.3.

OB HANA

V České Republice do roku 2012 nejpoužívanější systém pro přihlašování[18]. Nahrazen systémem ORIS. Webová aplikace s velmi zastaralým a nepřívětivým uživatelským rozhraním.

Orienteering Online

Mezinárodní přihlašovací systém. Webová aplikace. Export pouze do vlastních formátů CSV a XLS.

1.4.4 Systémy pro přípravu závodních dat

Pro přípravu závodních dat se dnes běžně používají grafické nástroje, které staviteli tratí dovolují tvořit tratě přímo v mapě. Systém pak z nakreslených tratí automaticky vytvoří data pro závod.

Orienteering Organiser

Český software pro podporu přípravy a organizace událostí (kapitola 1.5.3). Příprava závodních dat přímo na mapových podkladech s možností exportu připravených dat do mnoha formátů.

[15] http://www.4mila.com
[16] http://www.tak-soft.com/products/sport/sime/
[17] http://oris.orientacnisporty.cz
[18] http://www.obhana.cz/prihlasky.asp

OEvent

Zahraniční software pro podporu přípravy a organizace událostí (kapitola 1.5.3). Příprava závodních dat je možná pouze bez grafických podkladů.

OCAD

Primárně kartografický software specializovaný na OSP (kapitola 1.5.5) s rozšířením dovolujícím tvorbu závodních dat na vytvořených mapových podkladech. Možnost exportu do IOF Interface Standard 2.0.3 a 3.0.

1.4.5 Systémy pro podporu tvorby map

Systémy pro podporu tvorby map jsou kartografické systémy umožňující elektronickou kresbu a editaci mapových podkladů a exportu vytvořené mapy pro tisk.

OCAD

Profesionální kartografický systém pro kresbu map pro orientační běh. Umožňuje definici mapového klíče, podkladů a všech parametrů kreslené mapy. Komerční software, pouze pro systém Windows[19].

OpenOrienteering Mapper

Multiplatformní open source kartografický systém pro kresbu map pro orientační běh[20]. Funkčně shodná se systémem OCAD.

1.5 Příslušenství k zařízením s iOS

Zařízení se systémem iOS (dále jen iPhone) bude jádrem celého terminálu. Proberu tedy problematiku připojování externích periferií. Znalost samotného vývoje aplikací pro iOS považuji za preprekvizitu studia této práce.

1.5.1 Typy příslušenství dle připojení

K iPhone je možné externí příslušenství fyzicky připojit různými způsoby a pro komunikaci použít různé komunikační protokoly. Způsob připojení zařízení je určující pro volbu knihovny pro integraci periferního zařízení do iOS aplikací.

Nejjednodušší – a společností Apple preferovanou – možností je připojit periferní zařízení bezdrátově, tedy pomocí WiFi. Při připojení externího zařízení pomocí **WiFi** je možné využít jakoukoliv technologii pro síťovou komunikaci integrovanou v iOS, tedy například BSD sockets nebo HTTP/XML/JSON atp., či použít jakoukoliv rozšiřující knihovnu třetích stran s implementací specifického komunikačního protokolu například Pion HTTP server[21].

Další bezdrátovou možností je připojit externí příslušenství pomocí **Bluetooth**. Připojování takových externích zařízení svázáno přísnějšími pravidly. Vývojář externího zařízení musí komunikaci implementovat pomocí bluetooth profilu obsaženého v systémovém frameworku GameKit nebo se stát členem programu Apple MFi (viz. dále). Další možností je využití frameworku

[19] http://www.ocad.com/en/
[20] http://sourceforge.net/projects/oorienteering/
[21] https://github.com/cloudmeter/pion/wiki

Core Bluetooth, který je součástí iOS 5 a vyšších. Core Bluetooth podporuje pouze Bluetooth zařízení verze 4.0 Low Energy.

Speciálním případem jsou externí zařízení připojovaná přes **audio jack**. Tato zařízení komunikují s iOS aplikací prostřednictvím knihovny Core Audio[22]. Jedná se o analogovou komunikaci přes AD převodník v audio čipu iPhone. Nejznámější implementací je zařízení Square pro akceptaci platebních karet[23].

Poslední možností je využít k připojení externího příslušenství **30 pinový dokový** konektor nebo novější **Lightning** konektor. Ke komunikaci s externím zařízením se používá systémový External Accessory framework, který externí zařízení pro iOS aplikaci zapouzdřuje. Externí zařízení musí být vyvinuto účastníkem Apple MFi programu.

1.5.2 Apple MFi program

Apple MFi je program určený pro vývojáře externího příslušenství k iPhone, iPad a iPod touch připojitelném přes Bluetooth nebo dokový či Lightning konektor.

Účast v MFi programu je velmi omezena, žadatel musí splňovat mnoho kritérií pro zařazení a musí projít externím auditem. Žadatel musí být právnická osoba a musí předložit finanční zhodnocení společnosti. Externí audit je placený, provádí ho společnost pověřená společností Apple a náklady nese žadatel.

Pokud má žadatel v plánu navržené zařízení vyrábět a prodávat koncovým zákazníkům potřebuje navíc Manufacturing License. Pro výrobu pro koncové zákazníky je možné používat outsourcing. Dodavatel ale musí mít příslušnou licenci.

Veškerá dokumentace a jiné zdroje poskytnuté v rámci MFi programu podléhají přísné NDA (dohoda o mlčenlivosti) a účastníkům programu není dovoleno zdroje a dokumentaci jakkoliv zveřejňovat.

Vzhledem k těmto podmínkám je MFi program pro běžné fyzické osoby téměř nedosažitelný.

1.5.3 Redpark TTL kabel

TTL nebo RS-323 kabel od americké společnosti Redpark je volně dostupným produktem vyvinutým ve shodě s MFi licencí určeným pro připojování vlastních externích zařízení k iPhone přes sériovou linku.

Výrobce ke kabelům poskytuje SDK – po registraci[24] – s knihovnou „Redpark Serial Cable Manager“. Knihovna poskytuje rozhraní pro komunikaci s Redpark TTL/RS-323 kabelem a zapouzdřuje komunikaci s External Accessory frameworkem.

Knihovna obsahuje Objective-C třídu RscMgr. Instance této třídy slouží pro komunikaci s kabelem. Pro asynchronní komunikaci a notifikaci událostí používá návrhový vzor delegát. Instance třídy sloužící jako delegát instance RscMgr musí implementovat Objective-C protokol RscMsgDelegate. Další potřebné informace k integraci komunikace se sériovým kabelem do aplikace jsou přehledně dostupné v SDK poskytovaném výrobcem. Jako průvodce mohu doporučit knihu iOS Sensors Apps with Arduino [10].

Sériový kabel od firmy Redpark je kvalitním řešením pro vývoj vlastních externích zařízení pro iPhone bez nutnosti stát se členem MFi programu.

---

[22] Framework v iOS SDK pro práci s audiem. Dokumentace dostupná z http://developer.apple.com/library/ios/#documentation/MusicAudio/Conceptual/CoreAudioOverview/CoreAudioEssentials/CoreAudioEssentials.html

[23] https://squareup.com

[24] http://www.redpark.com

Obrázek 1.2 – Redpark sériový TTL kabel (http://www.redpark.com/c2ttl.html)

1.6 Hardwarové komponenty terminálu

V této kapitole teoreticky představím hardwarové komponenty použité při návrhu a vývoji terminálu. Uvedu i implementační detaily konkrétních typů použitých komponent.

1.6.1 Mikrokontroléry

Mikrokontrolér je integrovaný obvod umožňující obsáhnout celou aplikaci vestavěného systému bez podpory jiných obvodů. Mikrokontrolér kromě procesoru obsahuje operační paměť, paměť pro program, adresové, řídící a datové sběrnice a periferní jednotky. Důležitými parametry jsou šířka datové sběrnice, typ mikroprocesoru, velikost paměti či počet vstupní, výstupních, analogových vývodů či komunikačních sběrnic.

Architektura mikrokontroléru může spojovat paměť pro data a program, pak se jedná o architekturu Von Neumannovu nebo může být paměť pro data a program oddělena, pak se jedná o Harvardskou architekturu. V současné době se využívá obou přístupů.

Mikrokontrolér typicky obsahuje několik obecných vstupně/výstupních digitálních portů, několik portů A/D či D/A převodníku a sériové komunikační linky. Některé digitální porty obsahují přerušení, tedy možnost vykonání programátorem definované funkce při změně stavu portu.

Programátor při vytváření programu pro mikrokontrolér pracuje s jeho programovacím modelem, který definuje registrovou strukturu a sadu instrukcí. Program se vytváří v jazyku symbolických instrukcí. Většina výrobců mikrokontrolérů dnes dodává pro své výrobky překladače z jazyka C či C++ do jazyka symbolických instrukcí daného mikrokontroléru.

Další informace o mikrokontrolérech je možné nalézt například ve studijní opoře předmětu „Mikroprocesorové a vestavěné systémy“ [1].

Mikrokontroléry Atmel AVR

Rodina 8-bitových mikrokontrolérů s redukovanou instrukční sadou (RISC) a s Harvardskou architekturou od firmy Atmel. Program je uložen v trvalé flash paměti. Velikost paměti pro program je uvedena v kódovém názvu čipu. Veškerý program musí být uložen v této paměti a neexistuje způsob, jak část programu odložit jinam. Obsahují 32 jedno-bajtových registrů.

Mikrokontroléry AVR mají dvoufázové jednoúrovňové zpracování instrukcí. Následující instrukce je načtena při provádění instrukce aktuální. Většina instrukcí trvá pouze jeden nebo 2 cykly. AVR byly navrženy tak, aby velmi dobře pracovali s přeloženým kódem v jazyce C.

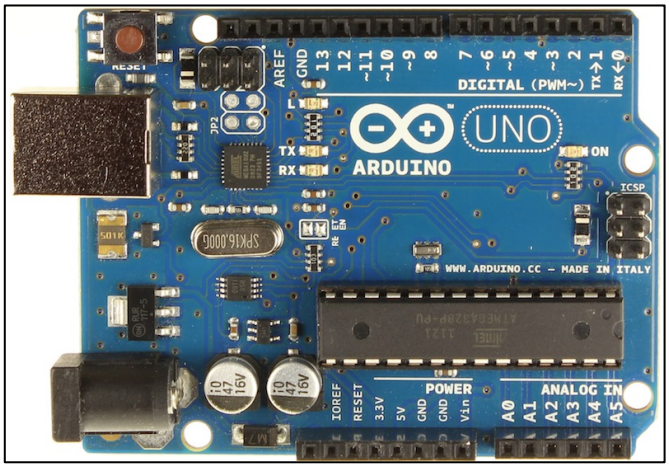
1.6.2 Vývojové kity Arduino

Arduino je open-source vývojová platforma sloužící k rychlému prototypování vestavěných systémů[25]. Arduino se skládá z vývojových kitů s procesory Atmel AVR Mega, multiplatformního integrovaného vývojového prostředí (IDE) a programovacího jazyka. IDE je vyvinut v jazyce Java, obsahuje editor zdrojového kódu, překladač a programátor kitů. Programovací jazyk je podobný C++, má ovšem nějaká omezení. Součástí platformy je i množství knihoven, jež je možno při programování využít. Jedná se například o knihovny Serial[26] a SoftwareSerial[27] sloužící ke komunikaci přes sériovou linku.

Existuje několik druhů kitů. Všechny kity mají podobné rozsazení vývodů a je tedy možné připojovat k nim další desky se specifickým zapojením a funkcí – takzvané „shields“. Existují speciální rozšiřující desky určené k prototypování. Tyto desky je možné zapojit do vývodů vývojového kitu a realizovat na nich vlastní obvod.

Vývojové kity lze k počítači připojit pomocí USB a pro komunikaci s okolím obsahují jednu nebo více UART sériových linek s úrovněmi TTL. Další sériové linky je možné simulovat přes univerzální vstupně-výstupní digitální vývody. Pro simulaci sériové linky musí být příslušný přijímací vývod (Rx) vybaven obsluhou přerušení.

Arduino kit může být napájen z USB portu externím stejnosměrným napětím 7-9V či stabilizovaným napětím 5V.

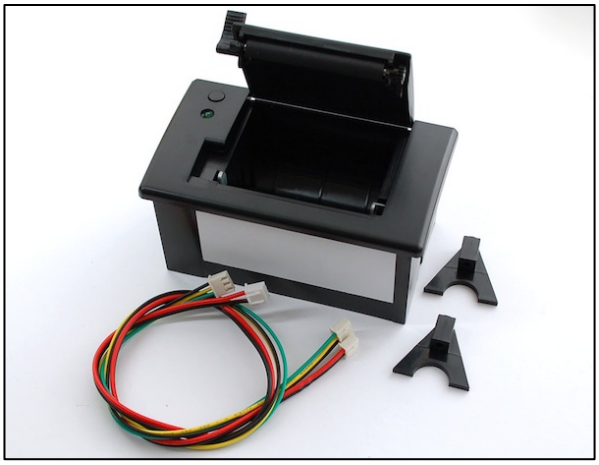


Obrázek 1.3 – Vývojový kit Arduino Uno Rev3 (http://arduino.cc/en/Main/ArduinoBoardUno)

1.6.3 Adafruit Mini Thermal Printer

Adafruit Mini Thermal Printer je kompletní modul miniaturní termální tiskárny. Modul je složen z tiskové hlavy, řídící desky vybavené sériovým rozhraní s TTL úrovněmi a platovým obalem pro zavedení a úschovu role papíru. Tiskárna tiskne metodou přímého termálního tisku (viz dále), obsahuje znakovou sadu ASCII a GB2312-80. Umožňuje základní formátování textu, tisknout čárové a QR kódy a vlastní monochromatické bitmapy. Pro komunikaci po sériové lince definuje vlastní protokol [11]. Adafruit k tiskárně dodává knihovnu pro kity Arduino umožňující tisknout bez znalosti samotného protokolu modulu. Ke kitu se tiskárna připojuje přes obecné digitální vývody s využitím knihovny SoftwareSerial.

[25] http://arduino.cc/
[26] http://arduino.cc/en/Reference/Serial
[27] http://sundial.org/arduino/index.php/newsoftserial/



Obrázek 1.4 – Adafruit Mini Thermal Printer (https://www.adafruit.com/products/597)

Přímý termální tisk

Metoda tisku, při které je jediným spotřebním materiálem papír. Tisková hlava je složena s miniaturních rezistorů s velmi malou teplotní setrvačností, jednotlivé body jsou do papíru vypalovány.

1.6.4 Asynchronní sériová komunikace

Asynchronní sériová komunikace je taková komunikace, při které se využívá pouze jeden datový vodič – případně 2 (pro každý směr komunikace jeden). Vodič pro přenos hodinového signálu se nepoužívá. Místo toho se před začátkem komunikace provede synchronizace vnitřních hodin přijímače a vysílače pomocí start-bitu. Komunikace se ukončí pomocí jednoho nebo více stop-bitů. Detailnější informace o principech asynchronní sériové komunikace je možné získat ve studijní opoře k předmětu „Mikroprocesorové a vestavěné systémy“ [1].

UART

Univerzální asynchronní přijímač/vysílač je zařízení, které umožňuje komunikovat prostřednictvím asynchronní sériové komunikace. Většinou je ve shodě s komunikačním standardem (například RS-232). Jednotlivé znaky jsou odesílány v rámcích (viz tabulka 1.3). Paritní bit je volitelný. Stop bitů může být více než jeden. Reálně stop bit znamená uvedení linky do klidového stavu před dalším vysíláním (start-bitem). Linka může být v klidovém stavu libovolně dlouhou dobu, ale delší než počet stop-bitů.

Rychlost přenosu je parametr, který musí být shodně nastaven na přijímači i vysílači, uvádí se v baudech nebo-li bitech přenesených za sekundu. Rychlost přenosu ovlivňuje takt vnitřních hodin přijímače i vysílače.

Start bit	Data bit 0	Data bit 1	Data bit 2	Data bit 3	Data bit 4	Data bit 5	Data bit 6	Data bit 7	Parity bit (opt)	Stop bit(s)

Tabulka 1.3 – UART rámec znaku

Napěťové úrovně

Normy sériových linek definují elektrické charakteristiky rozhraní. Jedním z nejdůležitějších parametrů je určení napěťových úrovní pro logickou 0 a 1. V práci budu kombinovat 2 úrovně napětí. TTL a RS-232.

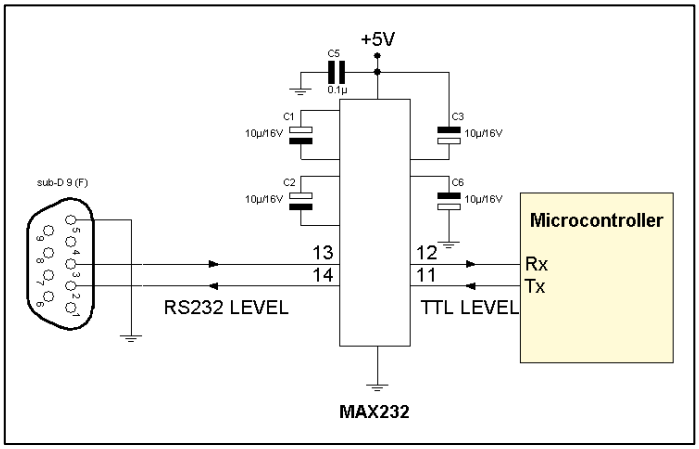
Obvody **TTL** (transistor-transistor-logic) využívají napětí 4,5 V – 5,5 V. Logická 1 je definována jako napětí větší než 2,0 V a menší nebo rovno 5 V. Logická 0 jako napětí větší nebo

rovno 0 V a menší než 0,8 V. Pásmo 0,8 V – 2,0 V je zakázané pásmo, ve kterém by se hodnota napětí na vstupních linkách neměla dlouhodobě vyskytovat.

Z historických důvodů rozhraní **RS-232** používá rozdílné úrovně napětí. Logická 1 je definována jako napětí v pásmu -15 V až -3 V, logická 0 jako napětí v pásmu 3 V až 15 V. Mezi těmito úrovněmi leží opět zakázané pásmo.

MAX-232

Pro převod napěťových úrovní se využívají speciální integrované obvody. Nejznámějším je asi obvod MAX-232 [12]. Jedná se o dvoukanálový, obousměrný převodník napěťových úrovní z TTL na RS-232 (viz obrázek 1.5).



Obrázek 1.5 – typické zapojení obvodu MAX-232 (http://arduino.cc/forum/index.php?topic=145382.0)

1.6.5 Napájení

Elektronické stabilizátory napětí

Stabilizátory napětí jsou elektrická zapojení umožňující stabilizovat výstupní napětí při změnách vstupního napětí, teploty či jiných relevantních veličin. Často jsou dodávány ve formě integrovaného obvodu. Způsobů zapojení existuje velké množství. O stabilizátorech blíže pojednává například Wikipedia [13].

Stabilizátor 78S05

Tří-vývodový pozitivní regulátor pro okamžitý odběr proudu do 2A. Eliminuje vstupní rušení. Vstupní napětí musí být minimálně 1V. Typicky se zapojuje jako konstantní stabilizátor napětí společně se 2 kondenzátory.

Integrované moduly správy napájení

Moduly pro správu napájení (PMIC) jsou integrované obvody pro řízení napájení vestavěného systému. PMIC se dnes vyskytují ve všech systémech s vestavěnou baterií – například v mobilních telefonech, fotoaparátech či mp3 přehrávačích). PMIC většinou spojuje funkce jako jsou: DC/DC konverze, nabíjení vestavěné baterie, výběr zdroje napájení či škálování napětí. Pro komunikaci s mikrokontroléry často obsahují nějakou formu sériové sběrnice. Výrobci PMIC jsou například Texas Instruments[28] či Linear Technology[29].

[28] http://www.ti.com
[29] http://www.linear.com

2 Návrh řešení

V první části této kapitoly se budu věnovat své vizi a řešení problému. Proberu své pohnutky k řešení problematiky a uvedu požadavky budoucích uživatelů (kapitola 2.1). V dalších částech kapitoly navržené řešení představím. Nejdříve se budu věnovat hardwarové části terminálu (kapitola 2.2), dále uvedu specifikaci rozšíření protokolu pro přenos razících dat (kapitola 2.3) a potom popíši softwarové části terminálu. Tedy Knihovnu EasyPunch (kapitola 2.4), jejíž součástí je i obslužný program mikrokontroléru a iOS aplikaci EasyEvent (kapitola 2.5).

2.1 Vize a cíl

Mojí motivací pro řešení problému v oblasti organizace závodů v OSP byla především potřeba přispět svým dílem komunitě orientačních běžců, jejíž jsem dlouholetým členem. Jako člověku, specializujícím se profesně na problematiku mobilních a internetových aplikací, uživatelský přínos a přívětivost, mi situace viditelná na každé události v OSP není příjemná.

Výpočetní technika používaná při organizaci událostí v OSP a zpracování výsledků je příliš složitá a těžkopádná. Díky tomu často vznikají problémy, které narušují hladký průběh události.

Z šetření mezi potenciálními uživateli mého řešení (kapitola 2.1.1) vyplynula jasná potřeba konsolidovat současnou výpočetní techniku používanou při organizaci závodů do menších a přenosnějších zařízení. Dále také potřeba zrychlení toku závodních dat, ať už v místě události, či mezi místem události a internetovými službami.

Problém jsem se rozhodl řešit postupně. Jak jsem uvedl v úvodní kapitole, za prvořadý cíl jsem zvolil návrh a tvorbu přenosného integrovaného terminálového řešení obsahující všechny důležité prvky výpočetní techniky i softwarového vybavení. Terminál by měl výrazným způsobem zjednodušit zpracování přihlášek, startovní listiny a výsledků. Zároveň by měl být pro organizaci závodů jediným nezbytným kusem výpočetní techniky. Mým druhořadým cílem se logicky stala příprava a zveřejnění multiplatformní Open Source knihovny EasyPunch a Protokolu Punch pro přenos razících a jiných závodních dat.

Dal jsem si za cíl celý systém navrhovat modulárně a maximálně otevřeně tak, aby jej bylo později možné doplnit o další funkce.

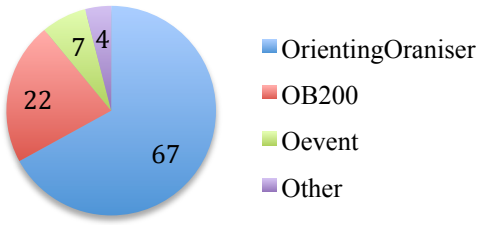
2.1.1 Výsledky šetření mezi uživateli

Před definicí cíle a řešením problému jsem provedl průzkum mezi potenciálními budoucími uživateli. V průzkumu jsem se zaměřil na 2 skupiny uživatelů. Organizátory a závodníky. Zástupcům skupin byly předloženy 3 části dotazníku. V první části bylo uvedeno 5 funkcí, které výpočetní technika zastává (tabulka 2.1). Respondenti měli za úkol oznámkovat je podle spokojenosti s nimi. Druhá část se týkala pouze organizátorů a zjišťovala, který současný nástroj pro organizaci závodů nejvíce používají. Výsledky jsem zpracoval do grafu 2.1. Třetí částí bylo interview s vybranými zástupci obou skupin. Každému respondentovi jsem postupně položil 3 otázky (tabulka 2.1), které jsem na základě odpovědí upřesňoval.

Z výsledků vyplývá, že organizátorům nevíce vadí mobilita komponent VT používaných při událostech, závodníkům na druhou stranu způsob prezentace výsledků po závodech. Oběma skupinám na druhou stranu nejvíce vyhovuje systém ražení SPORTIdent.

	Závodníci	Organizátoři
Funkcionality	Systém ražení (1,43)	Služby pro přípravu mapových podkladů (1,23)
	Závodní weby (2,94)	Správa systému ražení (2,11)
	Rychlost prezence a vyčtení (3,27)	Služby pro komunikaci se závodníky (2,42)
	Dostupnost výsledků (3,54)	Jednoduchost přípravy VT událostí (3,71)
	Prezentace výsledků (4,22)	Mobilita VT při události (4,56)
Otázky v interview	Jak jsou dostupné předzávodní informace?	Rozdělte činnosti při organizaci události dle pracnosti.
	Jaká služba vám nejvíce chybí po skončení závodů a jaká Vám naopak nejvíce vyhovuje?	Které části organizace událostí Vás nejvíce zatěžují? Co byste potřebovali zlepšit? (z pohledu výpočetní techniky)
	Co by jste zlepšili v organizaci závodů?	Jaká je Vaše vize komunikace se závodníky?

Tabulka 2.1 – Šetření mezi uživateli, zadání dotazníku s průměrným hodnocením funkcionalit



Graf 2.1 – Tržní podíl (%) nástrojů pro organizaci událostí v OSP

2.2 Návrh hardwarové části

V kapitole popisuji navržené řešení hardwarové části terminálu. Nejdříve se zaměřím na jednotlivé moduly terminálu. Dále popíši komunikační kanály mezi moduly a budu se věnovat napájení a obalu. V textu zdůrazním rozdíly mezi prototypem vytvářeným v rámci této práce a finálním terminálem.

2.2.1 Blokový model terminálu

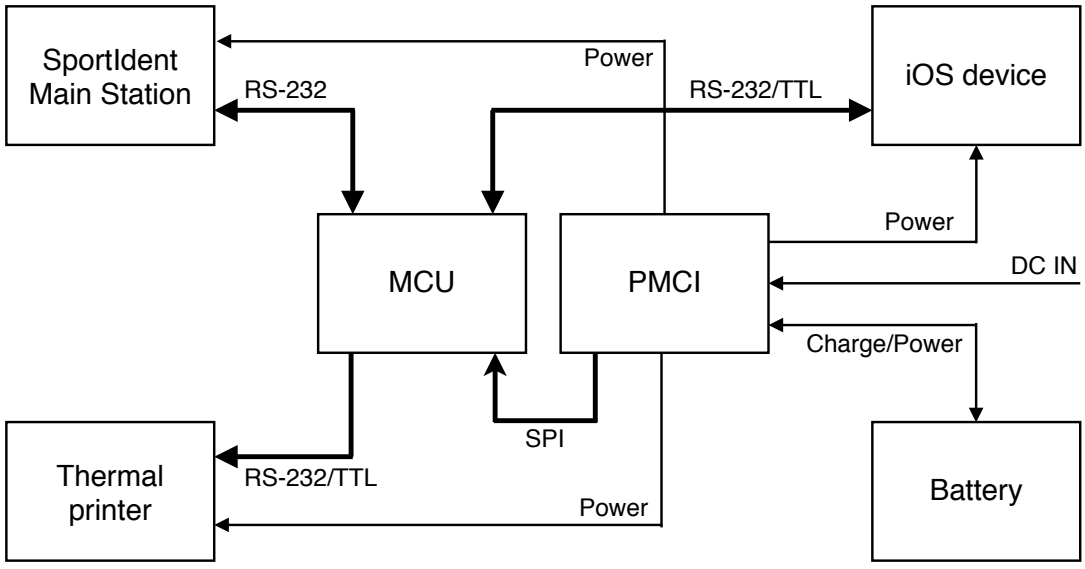


Schéma 2.1 – Blokové schéma terminálu

Terminál se skládá z modulů termální tiskárny (Adafruit Mini Thermal Printer), základové jednotky SPORTIdent BSx7, libovolného iOS zařízení s 30 pinovým dokovým konektorem, PMIC obvodu, Li-Ion jedno-článkového akumulátoru a mikrokontroléru Arduino Uno Rev 3. Moduly jsou propojeny datovou sběrnicí a napájeny pomocí akumulátoru a PMIC modulu.

2.2.2 Zapojení modulů

Elektrické zapojení obvodu terminálu jsem vyjádřil výkresem v příloze 2. Podrobnosti o zapojení napájecí části uvádím v kapitole 2.2.3. V dalším textu popisuji zapojení modulů s odkazy na součástky výkresu, které jsem uvedl v závorkách.

Základová jednotka SPORTIdent

Základová jednotka SPORTIdent BSx7 je dodávána s rozhraním USB nebo RS-232. Pro aplikaci v terminálu jsem z důvodů snazší implementace zvolil rozhraní RS-232. Jednotka je sériovou linkou připojena k GPIO portům Arduina 2 a 3 (JP4). Vzhledem k rozdílným napěťovým úrovním linky RS-232 a GPIO portům Arduina je mezi Jednotku a Arduino vložen převodník napěťových úrovní MAX-232 (IC1).

Na obalu prototypu terminálu je umístěn speciální konektor D-sub DE9 (PG-x) pro přímou komunikaci se zabudovanou jednotkou. K aktivaci této komunikace slouží přepínač S1.

Adafruit Mini Thermal Printer

Modul tiskárny je dodáván s integrovaným sériovým rozhraním s úrovněmi TTL (JP6). Je jej tedy možné sériovou linkou připojit přímo k GPIO portům 4 a 5 Arduina (JP4).

Modul zařízení se systémem iOS

iOS zařízení je připojeno pomocí svého 30 pinového dokového konektoru. Jak jsem uvedl v kapitole 1.6.1, pro dokový konektor není k dispozici veřejná specifikace. K převodu na sériovou linku je tedy použit Redpark TTL kabel, který pomocí integrované elektroniky převádí signály z dokového konektoru na sériovou linku s několika řídícími signály (JP1).

Sériová linka z Redpark TTL kabelu je připojena k sériovému portu Arduina (RX a TX vývody JP4). Sériová linka Arduina se využívá i pro programování mikrokontroléru. Aby bylo možné mikrokontrolér naprogramovat, je nutné odpojit Redpark TTL kabel. K tomuto účelu slouží přepínač (S3).

2.2.3 Napájecí obvod

O napájení terminálu se stará PMIC obvod. Do této chvíle se mi zatím nepodařilo nalézt obvod, který by vyhovoval aplikaci terminálu. Z tohoto důvodu jsem se rozhodl prototypový terminál sestavit bez PMIC obvodu a baterie.

Prototypový terminál bude napájen externím zdrojem stejnosměrného napětí 12 V. Maximální okamžitý odběr jsou 2 A. Základová stanice SPORTIdent a iOS zařízení budou v prototypu k napájení využívat svůj interní akumulátor. Arduino a modul tiskárny bude napájen stabilizovaným napětím 5 V, které poskytuje elektronický stabilizátor napětí L78S05 (IC2) v zapojení se 2 podpůrnými kondenzátory (C1, C2), signalizační LED diodou (D1, R1) a vypínačem (S2).

Výpočet kapacity baterie

Jako základní parametr pro výpočet potřebné kapacity baterie určuji požadovanou dobu provozu na baterii. Tento parametr jsem stanovil na 5 hodin. To by mělo být dostatečné i pro ty nejdelší závodní

události. Pro výpočet potřebné kapacity baterie je třeba znát průměrnou spotřebu jednotlivých modulů terminálu, včetně režijních nákladů na řízení napájení PMIC modulem.

Spotřeba základové stanice SPORTIdent je zanedbatelná. Výrobce do jednotky obsazuje AA Li-Ion akumulátor o kapacitě 1200 mAh a uvádí přibližnou dobu běžného provozu na tento akumulátor 5 let. Z toho důvodu tento modul ve výpočtu zanedbávám.

Spotřeba termální tiskárny je závislá na době používání. Tiskárna má maximální okamžitý odběr 1,5 A při 5 V. Tohoto odběru se však dosahuje jen velmi krátkodobě při tisku. Předpokládám, že tiskárna bude soustavně tisknout maximálně polovinu doby požadované výdrže na baterii, tedy 2,5 hodiny. Tiskárna má tedy průměrnou spotřebu 3,75 W.

Spotřeba iOS zařízení se velmi liší dle připojeného typu a aplikacích, které jsou na něm zapnuty. Například při zapnuté WiFi a GSM/3G modulu je spotřeba několikanásobně vyšší než při vypnuté bezdrátové komunikaci. Pro výpočet jsem použil hodnoty popsané v článku serveru AnandTech [14]. Relevantní je hodnota při zapnutém a používaném 3G modulu, tedy průměrnou spotřebu 2W.

Spotřebu mikrokontroléru (společně s převodníkem úrovní), jsem určil praktickým měřením. Mikrokontrolér jsem spustil s finálním programem a měřil proud, který odebírá. Při napětí 5 V odebíral konstantní proud 0,25 A. Průměrná spotřeba tedy je 1,25 W.

Režijní náklady PMIC modulu nejsem v tuto chvíli schopný určit, proto je ve výpočtu zanedbávám.

Celková průměrná spotřeba terminálu je tedy 7 W. Nominální napětí jedno-článkových Li-Ion akumulátorů je typicky 3,7 V. Při spotřebě 7 W musí mít akumulátor kapacitu 9,46 Ah. Vzorce použité pro výpočet jsou dostupné například na Wikipedii [15].

2.2.4 Obal a krytí

Samotný výsledný obal terminálu jsem doposud nevybral. Na obal kladu vysoké nároky, které se mi zatím nepodařilo splnit. Jde především o minimální krytí IP56 dle normy IEC60529 a ergonomické vlastnosti obalu. Po testování prototypového terminálu počítám s výrobou prototypu obalu pomocí 3D tisku.

Pro prototyp terminálu jsem použil univerzální platovou krabičku, do které jsem zabudoval všechny moduly.

2.3 Protokol Punch

Jak zmiňuji v kapitole 1.3.1 systém elektronického ražení SPORTIdent využívá ke komunikaci základové stanice s počítačem sériové rozhraní a vlastní komunikační protokol. Pro komunikaci mezi komponentami terminálu, ale i mezi samotnými terminály či jinými zařízeními je tento protokol nevyhovující. Mým záměrem bylo navrhnou rozšíření původního protokolu tak, aby umožňoval přenášet, kromě dat o ražení a nastavení kontrolních jednotek, i další typy dat, byl jednoduše rozšiřitelný a nebyl závislý na transportní vrstvě. Jedním z imperativů návrhu byla možnost řetězit různá zařízení za sebe a přenášet data různými cestami jedním protokolem. Například od kontrolní jednotky k terminálu po sériové lince a z terminálu na server v internetu přes WebSocket.

Z pohledu ISO/OSI či TCP/IP modelu se jedná o protokol aplikační vrstvy nezávislý na transportní vrstvě.

2.3.1 Módy komunikace

Po navázání spojení mezi klientem a hostitelem (viz kapitola 2.3.2) probíhá další komunikace v jednom ze tří módů dle typu hostitelského zařízení.

- mód **BSx6** definuje, že hostitelské zařízení je kompatibilní pouze s původním protokolem SPORTIdent verze 4.
- mód **BSx7** definuje, že hostitelské zařízení je kompatibilní s původním protokolem SPORTIdent verze 4 i 5.
- mód **Extended** definuje, že hostitelské zařízení umožňuje komunikaci pomocí jakékoliv verzí původního i nového protokolu.

Mód komunikace tedy definuje množinu i strukturu zpráv, kterým jednotlivé hostitelské zařízení rozumí.

2.3.2 Struktura zprávy

Zpráva je základní jednotka komunikace protokolu Punch. Zápis a čtení zprávy z či do komunikační linky zajišťuje transportní vrstva a s ní použitý protokol. Protokol Punch pouze definuje strukturu zprávy, tedy pořadí a počet zapisovaných bajtů.

Rámec a definice zprávy

Každý mód komunikace využívá jinou strukturu zprávy. Strukturu zprávy pro módy BSx6 a BSx7 popisuje dokumentace výrobce systému SPORTIdent. Struktura zprávy pro rozšířený (Extended) mód je navržena tak, aby nebyla se strukturou zpráv původního protokolu konfliktní.

STX 0x02	0x00	Délka kódu příkazu (1B)	Kód příkazu	Délka těla zprávy (2B)	Tělo zprávy	CRC (2B)	ETX 0x03

Tabulka 2.2 – Rámec zprávy protokolu Punch

Délka zprávy je proměnlivá a vypočítá se součtem délky kódu příkazu a délky těla zprávy a přičtením 8 bajtů (viz tabulka 2.2). Maximální délka zprávy je tedy 65288 bajtů. Vzhledem k rychlosti sériové komunikace je doporučeno zasílat zprávy do délky 512 bajtů. Případné rozdělení dat zprávy do více stránek musí definovat příslušná zpráva ve své vlastní struktuře.

Přiřazení kódu příkazu nově definované zprávě se řídí pravidly uvedenými v tabulce 2.3. Rezervované kódy jsou určeny pro obslužné zprávy definované autorem protokolu Punch. Kódy zpráv delší než 2 bajty jsou v každé délce rozděleny do 3 pásem. Kratší kódy by měly být používány pro častěji zasílané zprávy.

Kód zprávy od	**Kód zprávy do**	**Určení**
0x00	0xFFFF	Zprávy rezervované
$0x0100(00)^{+}$	$0x7FFF(FF)^{+}$	Zprávy definované v rámci registrovaných tříd
$0x8000(00)^{+}$	$0xEFFF(FF)^{+}$	Zprávy definované v rámci vlastních tříd
$0xF000(00)^{+}$	$0xFFFF(FF)^{+}$	Zprávy definované mimo třídy

Tabulka 2.3 – Pravidla pro přidělování kódů zpráv v protokolu Punch

Kontrolní součet zprávy

Předposlední 2 bajty každé zprávy obsahují kontrolní součet (CRC). Kontrolní součet se počítá přes všechny bajty zprávy až k bajtům kontrolního součtu. Způsob výpočtu kontrolního kódu je definován v příloze 1. Při komunikaci v rozšířeném módu, se všechny bity součtu invertují. Příjemce zprávy

musí po přijetí zprávy vypočítat její kontrolní součet a porovnat jej se součtem přijatým. Pokud součty nejsou stejné odpoví příjemce odesílateli v případě rozšířeného módu zprávou „chybný kontrolní součet“ (viz tabulka 2.4) a v případě jiných módů „negativním handshake“ (viz dokumentace výrobce systému SPORTIdent).

STX 0x02	0x00	0x01	0x??	0x00	CRC (2B) 0x????	ETX 0x03

Tabulka 2.4 – Zpráva chybný kontrolní součet.

2.3.3 Struktura komunikace

Komunikaci zahajuje vždy klient. Klient nejdříve nastaví mód komunikace a zjistí třídu (třídy) hostitele. V případě úspěšného navázání komunikace klient i hostitel zasílají jednotlivé zprávy.

Transportní vrstva

Zprávy protokolu Punch mohou být zasílány přes libovolnou transportní vrstvu, která podporuje asynchronní duplexní přenos bajtů mezi klientem a hostitelem.

Speciálním případem transportní vrstvy je sériová linka RS-232. Komunikace v módu BSx6 a BSx7 může probíhat výhradně s touto vrstvou. Pokud protokol detekuje jakoukoliv jinou transportní vrstvu, musí být použit rozšířený mód.

Protokol Punch neobsahuje žádné prostředky pro autentizaci či autorizaci komunikujících stran. Tento úkol je možné ponechat na transportní vrstvě, či vytvořit speciální sadu zpráv podporující tyto vlastnosti.

Host – klient

Každé spojení má vždy 2 účastníky. Hostitele a klienta. Klient navazuje komunikaci a určuje její mód. Hostitel poskytuje klientovi informace o své třídě či třídách a odpovídá na zprávy klienta. Hostitel o samotném klientovi žádné informace nemá. Hostitel může klientovi zaslat zprávu bez předchozího vyžádání. Typicky se tak stane při externí události jako je například vložení čipu do základové jednotky. Klient takovou zprávu musí přijmout, ale pokud ji neumí obsloužit, může ji zahodit. Vzhledem k zaručené zpětné kompatibilitě s původním protokolem SPORTIdent může hostitel zasílat i automatické zprávy rozšířeného módu, i když klient vede komunikaci v některém z nižších módů.

Navázání spojení

Před začátkem komunikace musí klient s hostitelem navázat spojení a zvolit mód komunikace. Předpokladem pro navázání spojení protokolem Punch je úspěšné navázání spojení příslušnou transportní vrstvou. Tyto 2 procesy jsou nezávislé. Je tedy například možné, aby spojení na transportní vrstvě inicioval hostitel (například při připojení vysílače razicích dat přes WebSocket k serveru v internetu).

Navázání spojení je vícekrokové a navržené, tak aby bylo možné spolehlivě komunikovat i s jednotkami, podporujícími pouze původní protokol SPORTIdent. Původní protokol nebyl oddělen od transportní vrstvy. V případě, že je transportní vrstvou sériová linka RS-232 je třeba před začátkem navazování spojení nastavit baud rate na 38400 baud/s a dále musí být implementace protokolu Punch schopny nastavit v průběhu navazování spojení baud rate na 4800 baud/s. V případě, že se nejedná o sériovou linku RS-232, lze navázat pouze komunikaci v rozšířeném módu a jakékoliv parametry transportní vrstvy jsou pro protokol Punch irelevantní.

Prvním krokem navázání spojení je zaslání zprávy Get M/S mode s inverzními bity kontrolního součtu tak, aby formát zprávy odpovídal rozšířenému módu komunikace. Pokud hostitel odpoví kladně, je navázána komunikace v rozšířeném módu a proces navazování komunikace je u konce. Pokud hostitel odpoví negativním handshake či neodpoví vůbec, existují 2 možnosti. Pokud je transportní vrstva sériová linka RS-232, pokračuje se v navazování způsobem popsaným v dokumentaci výrobce systému SPORTIdent, v opačném případě se spojení nepodařilo navázat.

Zasílání zpráv

Zprávy je možné zasílat ihned po úspěšném navázání spojení. Zprávy jsou asynchronní a zasílat je může klient i hostitel. Odpovědí na zprávu může být pouze další zpráva zaslaná v opačném směru. Je na implementaci protokolu či na uživatelské aplikaci, aby tyto zprávy párovala, případně hlídala časový limit pro příjem odpovědi před opakováním dotazu. Protokol Punch nezaručuje žádné pořadí zasílání zpráv.

2.3.4 Třídy zařízení

Třída zařízení definuje seznam zpráv, kterým zařízení rozumí. Každé hostitelské zařízení musí podporovat zprávy jedné či více tříd.

Protože původní protokol SPORTIdent žádné třídy zpráv nepodporoval, jsou definovány 2 speciální třídy, do kterých jsou zařízení podporující pouze původní protokol zařazeny. Tyto speciální třídy jsou svázány s módem komunikace je zakázáno, aby je při komunikaci používal hostitel, který komunikuje s klientem přes rozšířený mód (viz dále).

V rozšířeném módu komunikace je možné zjistit třídy zařízení zasláním zprávy (viz tabulka 2.5). Každá třída je definována hexadecimálním kódem délky 4 byte. Celkem tedy muže být definováno více než 16 milionů tříd. Hodnoty 0x00000001 – 0x7FFFFFFF jsou rezervovány pro veřejně registrované a definované třídy. Hodnoty 0x80000000 – 0xEFFFFFFF jsou určené k vlastní libovolné implementaci. Zbylé hodnoty jsou rezervované a nesmí být použity pro definici třídy. Veřejně dostupné zařízení by měly vždy používat pouze registrované třídy.

Klient se může dotázat na třídu hostitelského zařízení zprávou s kódem 0x90 (viz tabulka 2.5), klient uvede pozici (POS) v seznamu tříd hostitele, kterou chce znát. Pokud klientovi není doposud, kolik má hostitel tříd uvede POS 0x00. Odpovědí od hostitele mu je zpráva s kódem 0x91 (viz tabulka 2.6). Hostitel v ní zopakuje pozici třídy v seznamu (POS) a zašle celkový počet tříd v seznamu (CNT) a kód samotné třídy na poptávané pozici. Z definice zprávy je jasné, že hostitel může mít pouze 256 zpráv. Pozice jsou číslovány od 0. V případě dotazu za hranici pole vrátí hostitel hodnotu kódu třídy 0x00000000.

STX				POS	CRC (2B)	ETX
0x02	0x00	0x01	0x90	0x??	0x????	0x03

Tabulka 2.5 – Zpráva pro dotaz na třídu zařízení

STX				POS	CNT	Kód třídy (4B)	CRC (2B)	ETX
0x02	0x00	0x01	0x91	0x??	0x??	0x????????	0x????	0x03

Tabulka 2.6 – Zpráva pro odpověď na dotaz na třídu zařízení

V přílohách 3 a 4 je možné nalézt definice tříd BSx6, BSx7, BSx6e, BSx7e a Mini Thermal Printer použitých v terminálu.

2.4 Knihovna EasyPunch

Knihovna EasyPunch je multiplatformní implementace Punch protokolu s podporou různých transportních vrstev, s možností definovat vrstvu vlastní. Implementace dále přináší podporu pro definici vlastních zpráv zasílaných protokolem Punch.

Jako implementační jazyk knihovny EasyPunch jsem z důvodu kompatibility s co nejširším počtem různých platforem zvolil jazyk C s použitím co nejmenšího počtu knihoven. Implementace je maximálně platformě nezávislá a přenos na jiné platformy by neměl být vůbec složitý. Platformě závislé jsou pouze moduly transportních vrstev, což ale vzhledem k často rozdílnému přístupu jednotlivých platforem ke komunikačním linkám není příliš na škodu.

2.4.1 Blokový model knihovny

Knihovnu EasyPunch je možné rozdělit na několik logických celků. Ústředním prvkem je Spojení, které spojuje modul transportní vrstvy s klientem, pomocí Parseru provádí překlad (serializaci/deserializaci) zpráv a zapisuje je resp. čte je z transportní vrstvy. Modul Parseru využívá k serializaci/deserializaci jednotlivých zpráv definice poskytované modulem Definice zprávy. Klient je potom klientská aplikace, která vytváří Spojení a pomocí struktury v každé definici zprávy vytváří nové a čte příchozí zprávy. Transportní modul slouží pro čtení a zápis jednotlivých bajtů z/do transportní vrstvy.

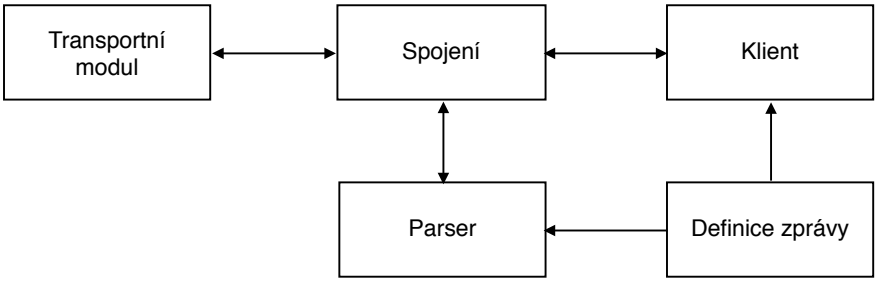


Schéma 2.2 – Blokový model knihovny EasyPunch

2.4.2 Zpráva

Každá zpráva je v knihovně EasyPunch definována strukturou zprávy, serializační a deserializační funkcí. Knihovna je dodávána se základní sadou zpráv. Uživatel knihovny může definovat zprávy vlastní.

```
typedef struct {
    const int8_t originalCode;
    const int64_t newCode;
} EPMessage;
```

Základem každé zprávy je struktura EPMessage. Všechny struktury zpráv musejí obsahovat strukturu EPMessage jako svoji první položku. Za touto položkou již následuje výčet parametrů konkrétní zprávy.

```
EPMessage* ep_message_alloc(int8_t aOriginalCode, int64_t aNewCode, EPError** aError);
void ep_message_free(EPMessage** aMessage);
```

Zasílané a přijímané zprávy musejí být alokované a uvolňované dynamicky pomocí funkce alloc a free. Funkce alloc podle předaných parametrů aOriginalCode a aNewCode automaticky zavolá příslušný alokátor pro danou zprávu a nastaví kód zprávy ve struktuře EPMessage. Vrácený ukazatel na strukturu typu EPMessage je ve skutečnosti ukazatel na strukturu alokované zprávy (tedy EPMessage i parametrů) a měl by být ihned po návratu přetypován na příslušný typ.

Funkce free uvolní všechny dynamicky alokované parametry (datové bloky či textové řetězce), uvolní samotnou strukturu zprávy a ukazatel nastaví na NULL. Parametrem funkce free je ukazatel na ukazatel na strukturu EPMessage. Ve skutečnosti se bude vždy jednat o ukazatel na ukazatel na konkrétní zprávu. Aby se předešlo varováním překladače bude vždy nutné provést přetypování.

Definice a registrace zprávy

Všechny typy zpráv, které chce klient přijímat či odesílat musí v knihovně EasyPunch zaregistrovat. Registrace se provádí globálně pro celý program. Každá zpráva je definovaná originálním kódem – tedy druhým znakem v rámci zprávy, novým kódem – tedy třetím a případně dalšími znaky. Jeden z těchto parametrů musí být 0, nebo se kód zprávy považuje za neplatný a registrace se nezdaří.

```
void ep_message_register(int8_t aOriginalCode, int8_t aNewCode,
                         EPMessage* (*aAllocator)(void),
                         void (*aSerialize)(EPMessage* aMessage,
                                            int8_t** aBuffer,
                                            int64_t* aLength, EPError**),
                         EPMessage* (*aDeserialize)(int8_t* aBuffer,
                                                    int64_t aLength, EPError**),
                         EPError** aError);
```

Dále pak alokátorem, tedy alokační funkcí, která dynamicky vytvoří strukturu zprávy a konečně serializační a deserializační funkcí, která předanou zprávu převede z či do pole bajtů odeslatelných po transportní vrstvě.

Serializer/deserializer

Serializační a deserializační funkce zprávy jsou odpovědné za správnou interpretaci parametrů zprávy. Pro různé platformní porty je nutné zajistit, aby se správně interpretovalo pořadí bitů i zarovnání struktur.

2.4.3 Spojení

Spojení je v knihovně EasyPunch implementováno jako abstraktní datový typ. Jedná se o neprůhlednou strukturu a operace definované nad touto strukturou.

```
typedef struct EPLink; // struktura je neprůhledná
```

Samotné spojení se může nacházet v několika stavech. Odpojeno, připojování a spojeno. Názvy jednotlivých stavů jsou sebevysvětlující.

```
typedef enum {
    EPLinkStateUnknown = -1,
    EPLinkStateDisconected = 0,
    EPLinkStateConnecting = 1,
    EPLinkStateConnected = 2,
} EPLinkState;
```

Existují 2 různé druhy otevření spojení podle toho, zda je zařízení klient či hostitel. Hostitel se nepřipojuje, pouze otevře transportním modulem kanál a vyčkává na připojení klienta (funkce listen). Klient se aktivně snaží navázat spojení s hostitelem (funkce open).

Vytvoření spojení

```
void ep_link_open(EPLink** aLink, char* aURL, void (*commandRec)(EPLink*, EPCommand*),
                  void (*aStatusClb)(EPLink*, EPLinkState, EPError*),
                  EPLinkDictionary* aOptions, EPError** aError);
```

Při otevření spojení klientem je třeba zavolat funkci open a předat jí parametry pro připojení. Existují 2 možnosti volby transportního modulu. Pokud je transportní modul zaregistrován (viz dále), předá se v parametru aLink neinicializovaný ukazatel na ukazatel a v parametru aURL adresa URL hostitele. Schéma v této adrese automaticky identifikuje transportní modul, který se použije. Schéma com:// označuje sériový port na PC, schéma ws:// transportní modul založený na technologii WebSocket. Pokud není možné podle schématu najít příslušný transportní modul, je vrácena chyba v parametru aError. Parametry aURL a aOptions jsou při otevírání spojení předávány dále do transportního modulu.

V případě potřeby zvolit pro komunikaci neregistrovaný modul, je třeba manuálně vytvořit a nakonfigurovat strukturu EPLink, tedy speciálně její zpětná volání transportního modulu. Odkaz na takovou strukturu je nutné předat při volání funkce open. Parametr aURL je v takovém případě irelevantní. Funkce open doplní klientské zpětné volání do struktury EPLink a dále se pokusí navázat spojení stejně, jako v prvním případě.

Proces připojování je popsán v kapitole 2.3.3 a knihovna EasyPunch se tímto popisem plně řídí. Funkce open je asynchronní, takže po zavolání ihned vrací. Funkce vytvoří pro připojování separátní vlákno. Klient musí zajistit běh procesu po dobu připojování, v opačném případě se připojení nevrací. V případě, že cílová platforma nepodporuje více vláknové zpracování, chová se funkce open synchronně a zpětné volání aStatusClb je voláno ještě před návratem z funkce.

V parametru aOptions je možné předat dodatečné parametry, například potřebné parametry pro připojení modulu transportní vrstvy jako je uživatelské jméno a heslo. Jednotlivé parametry definuje každý modul zvlášť.

Parametr commandRec je zpětným voláním, které se využívá k informování klienta o obdržení nové zprávy od hostitele.

Vyčkávání na spojení

```
void ep_link_listen(EPLink** aLink, char* aURL, void (*commandRec)(EPCommand*, EPLink*),
                    EPLinkDictionary* aOptions, EPError** aError);
```

Funkce link je využívána hostitelem pro vyčkávání na příchozí spojení. Parametry mají stejný význam jako v případě funkce open. Funkce listen je na rozdíl od funkce open synchronní a tedy blokující. Aktuální vlákno je zablokováno až do příchodu požadavku spojení od klienta.

Stejně jako u funkce open, je možné i u funkce listen vytvořit vlastní EPLink strukturu a otevřít spojení přes neregistrovaný modul transportní vrstvy.

Odeslání zprávy

```
void ep_link_send_command(EPLink* aLink, EPCommand* aCommand, EPError** aError);
void ep_link_flush(EPLink* aLink);
bool ep_link_isDone(EPLink* aLink);
```

Zprávu je možné odeslat pouze do spojení, které je ve stavu připojeno. V opačném případě je vrácena chyba. Pro odeslání zprávy se používá funkce send_command. Samotná zpráva je předána jako parametr aCommand.

Funkce je asynchronní a vrátí ihned po zařazení zprávy do odesílací fronty. Tedy je možné, že ještě před samotným odesláním zprávy. Pokud se chce uživatel ubezpečit, že před uzavřením spojení, či před ukončením klientské aplikace došlo k odeslání všech zpráv ve vnitřní frontě spojení, musí zavolat před uzavřením spojení funkci flush. Funkce flush vyčká na odeslání všech zpráv zbývajících ve frontě na transportní vrstvu a vrátí. Pokud platforma neumožňuje více vláknové zpracování, vrací funkce send_command také ihned po zařazení zprávy do fronty, jen se místo funkce flush používá funkce isDone. Funkce isDone vrací true, pokud je fronta prázdná. Mezi voláními funkce isDone musí být spojení dána příležitost data odeslat, typicky voláním metod modulu transportní vrstvy.

Před odesláním zprávy na transportní vrstvu se provede její serializace. Serializaci obstará modul parseru a serializační funkce dodaná při registraci zprávy. Samotná zpráva předaná funkci send přechází do vlastnictví spojení, které ji po odeslání automaticky uvolní – včetně jejích parametrů. Pokud má uživatel se strukturou zprávy další záměr, musí před jejím odesláním vytvořit její kopii.

Příjem zprávy

Pokud je dokončen proces přijetí zprávy, je zavolána klientská zpětná funkce commandRec poskytnutá klientem při otevření spojení. Pro identifikaci zprávy je součástí parametrů volání zpětné funkce ukazatel na strukturu spojení. Takto předaná zpráva přechází do vlastnictví klienta, který musí zajistit její pozdější uvolnění a to včetně dynamicky alokovaných parametrů.

Příchozí bajty jsou z modulu transportní vrstvy zasílány modulu parseru. Jakmile parser detekuje možný začátek zprávy a následně konec zprávy, předá vzniklý úsek dat deserializační funkci vybrané dle kódu zprávy. Pokud není příslušný kód zprávy zaregistrován nebo deserializaci selže, je celá zpráva zahozena. V případě úspěšné deserializace je volána klientská zpětná funkce.

Uzavření spojení

```
void ep_link_close(EPLink** aLink, EPLinkDictionary* aOptions, EPError** aError);
```

Spojení se uzavírá funkcí close. Funkce provede uzavření spojení. Modulu transportní vrstvy jsou při uzavírání předány parametry z aOptions. Uzavření spojení je okamžité, pokud má spojení ve frontě nějaké neodeslané či částečně neodeslané zprávy, nebudou tyto zprávy odeslány.
Pokud klient neprovede uzavření spojení, liší se následky podle použité transportní vrstvy. Některé transportní vrstvy ani uzavření spojení nepodporují.

V případě pokusu odeslat zprávu po uzavření spojení, je ohlášena chyba.

Samotná struktura spojení je automaticky uvolněna. Pouze v případě, že byla vytvořena klientem, musí ji klient sám uvolnit.

Více-vláknové zpracování a vlastnictví instance spojení.

Spojení je ve skutečnosti obousměrný tok dat. Hlavní úloha spojení – tedy kromě udržení aktivního spojení s protistranou – je serializace a deserializace zpráv. Spojení se nestará o správu procesů, běhovou smyčku a s určitými výjimkami ani o správu vláken. Je především na klientovi, ale také na modulech transportní vrstvy, aby zajistily více vláknové či událostmi řízené zpracování.

Klient musí po celou dobu, co používá spojení, vlastnit jeho instanci.

2.4.4 Moduly transportní vrstvy

Modul transportní vrstvy slouží pro čtení a zápis serializovaných zpráv z resp. do transportní vrstvy. Modul je z pohledu knihovny EasyPunch neprůhledná datová struktura implementující předepsanou množinu funkcí.

Při pokusu o navázání spojení vybere příslušná funkce Spojení inicializační funkci transportního modulu. Tato funkce se pokusí alokovat instanci modulu a vrátí na ni ukazatel. Instanci od té chvíle vlastní Spojení a je odpovědné, za její správné uzavření. Dále se Spojení pokusí otevřít kanál pomocí příslušné otevírací funkce modulu, které předá uživatelem zadané URL a další parametry.

Zápis dat do modulu ze Spojení se provádí voláním příslušné zapisovací funkce modulu. Čtení naopak invokuje modul zavoláním čtecí funkce Spojení.

Zavírací funkce uzavře kanál modulu a uvolní instanci modulu. Další funkcí je možné nastavovat rychlost sériové linky. Tuto funkci implementují pouze moduly RS-232.

Implementace a registrace modulu

```
void ep_link_register_module(char* aSchema,
                             void* (*aInitFunc)(int (*readFunc)(void*, int8_t, int),
                                                void* readFuncOpaque),
                             int (*aWriteBytes)(void* aModul, int8_t aBuffer,
                                                int aLength, bool aEndOfFrame),
                             bool (*aOpenFunc)(void* aModul, char* aURL,
                                               EPLinkDictionary* aOpts, EPError**),
                             void (**aCloseFunc)(void* aModul),
                             void (*setBdrt)(void* aModul, EPLinkBaudrate aBaudrate),
                             EPError**);
```

Každý modul, který má být při pokusu o spojení automaticky vybrán schématem URL, je nutné před jeho použitím registrovat v globálním registru modulů. Registrace se provádí funkcí register_module. Základním parametrem je textový řetězec se schématem, kterým se modul identifikuje. Ke schématu se registrují inicializační, otevírací, zápisová a uzavírací funkce a volitelně i funkce nastavující rychlost sériové linky.

Prvním parametrem všech funkcí je ukazatel na modul, který alokovala inicializační funkce. To je ve skutečnosti neprůhledná instance modulu.

Asynchronní zpracování příchozích zpráv

Pro příjem dat z transportní vrstvy nemá knihovna EasyEvent žádný pevně daný předpis nebo mechanismus. Je to ponecháno na odpovědnosti jednotlivých transportních modulů. Obecně ale doporučuji vytvořit pro čtení dat z transportní vrstvy samostatné vlákno a v něm provádět čtení a následný zápis do Spojení. Pokud platforma neumožňuje více vláknové zpracování, nesmí modul transportní vrstvy blokovat program čekáním na příchod zprávy. V takovém případě by měl používat nějaký vlastní mechanismus pro dotazování na dostupnost dat.

RS-232 modul

S knihovnou EasyPunch je dodáván jediný modul transportní vrstvy – modul pro komunikaci po sériové lince RS-232. Tento modul jako jediný dovoluje komunikovat v módu BSx6 a BSx7 a tedy jako jediný dovoluje nastavit rychlost komunikace sběrnice, pomocí speciální funkce.

Modul je implementován pomocí standardní POSIX knihovny termios a měl by být jednoduše použitelný na všech POSIX kompatibilních operačních systémech.

I když v navrhovaném prototypovém terminálu nebude tento modul nakonec použit, zařadil jsem jej pro testovací účely a možnost použití knihovny i mimo projekt terminálu.

2.4.5 Platformní porty

Aby bylo možné použít knihovnu EasyPunch v navrhovaném terminálu, bylo nutné ji přenést na 2 různé platformy – Arduino a iOS. V žádné z těchto platforem nelze použít modul transportní vrstvy RS-232. Ostatní implementační detaily knihovny, ale zůstávají téměř nezměněny.

Arduino Port

Program mikrokontroléru vývojového kitu Arduino je z velké části právě knihovna EasyPunch doplněná o vlastní transportní modu, modul obsluhující tiskárnu a běhovou smyčku.

Mikrokontrolér přijímá zprávy od iOS zařízení po sériové TTL lince, ty vyhodnotí a přepošle dále jednotce SPORTIdent nebo použije tiskovou třídu pro obsluhu tiskárny. V opačném směru zprávy přijaté od jednotky SPORTIdent přepošle iOS zařízení.

Knihovna EasyPunch se využívá jak ke klientskému spojení (jednotka SPORTIdent), tak k hostitelskému spojení (iOS zařízení).

Modul transportní vrstvy je jednoduchá třída postavená na zapouzdření třídy Serial a Software serial pro komunikaci přes sériovou linku dodávanou výrobcem systému. Tisková třída je pro Arduino dodávána výrobcem tiskárny.

Arduino nepodporuje více vláknové zpracování, proto je nutné neustále instance tříd Serial a Software serial dotazovat, zda-li nejsou k dispozici na přijímací straně nová data. Z těchto důvodů je v třídě modulu transportní vrstvy implementována funkce provádějící tento dotaz a případné následné čtení dat. Tyto funkce se pravidelně volají v běhové smyčce. Obecná posloupnost vykonávání programu vypadá tedy následovně: Obsluha došlých zpráv, vytvoření nových a jejich případné zaslání -> čtení dat ze sériového portu jednotky SPORTIdent -> čtení dat ze sériového portu iOS zařízení -> nový cyklus.

iOS Port

K Redpark TTL kabelu je dodávána sada Objective-C tříd pro sériovou komunikaci. Modul transportní vrstvy implementovaný v iOS aplikaci tedy zastřešuje tyto třídy. iOS aplikace jsou řízené událostmi a podporují více vláknové zpracování. Modul transportní vrstvy tedy pro čtení dat ze sériové linky využívá návrhového vzoru delegát, který nabízí třída dodávaná firmou Redpark.

Pro hladkou implementaci knihovny EasyPunch na klientské straně byl implementován obal nad spojením a zprávami v jazyce Objective-C. Díky tomu bylo možné zpětné funkce nahradit delegací a rozdílný formát zpráv dědičností.

2.5 EasyEvent Aplikace

V kapitole se věnuji designu obslužné aplikace terminálu – EasyEvent. Nejdříve definuji jednotlivé skupiny uživatelů a případy užití pro jednotlivé skupiny (kapitoly 2.5.1), abych později navrhl uživatelské rozhraní a uživatelskou logiku aplikace (kapitola 2.5.2). V další části navrhuji datový model aplikace a jeho konkrétní implementace pomocí technologie Core Data (kapitola 2.5.3). V poslední části se věnuji integraci datového modelu, uživatelského rozhraní, síťové komunikace a obslužné knihovny EasyPunch na aplikační úrovni (kapitola 2.5.4).

Díky zvolené platformě kapesního počítače a dodávaným SDK je velmi jednoduché aplikační funkcionality terminálu rozšířit třetími stranami. Kvůli aplikaci EasyEvent by ale potřeba rozšíření měla být výrazně minimalizována.

Aplikace EasyEvent je obslužná aplikace terminálu poskytující veškerou potřebnou funkcionalitu, kterou vyžaduje pravý dolní blok schéma 1.1 – Časomíra & výsledky. Aplikace umožňuje importovat vstupní data jednotlivých závodů – seznamy přihlášených, startovní listinu, informace o tratích a kategoriích či seznamy SI čipů k zapůjčení, provádět s daty operace potřebné během závodu – například startovat závodníky, číst obsah SI čipů, či provádět změny přihlášek a dohlášky – a exportovat výsledkovou listinu či změněné sady závodních dat.

Aplikace není určena pro předzávodní přípravu. Kreslení map, přípravu tratí či přihlašování závodníků je třeba provádět pomocí jiných nástrojů. Aplikace umožňuje organizaci závodu jednotlivců, nezaměřuje se na týmové či štafetové závody. Každý závod je separátní událost, zpracování více-etapového závodu lze pomocí aplikace provádět s pomocí jiného nástroje.

Import a Export dat z Aplikace lze provádět pomocí standardních XML formátů (viz kapitola 1.4.1). V případě potřeby použití většího počtu terminálů během jednoho závodu je možné aplikaci bezdrátově pomocí počítačové sítě propojit s jiným terminálem. Aplikace potom mezi sebou sdílejí datový model a pracují nad stejnou sadou dat.

2.5.1 Cílové skupiny uživatelů a jejich potřeby

Aplikace EasyEvent je navržena především s ohledem na její budoucí uživatele a jejich potřeby. Účelnému, jednoduchému a intuitivnímu návrhu uživatelského rozhraní předchází detailní poznání uživatele. Uživatele aplikace je možné rozdělit do několika skupin. Každá skupina bude aplikaci používat k různým operacím a bude vyžadovat jinou strukturu zobrazovaných informací.

V následujících podkapitolách popíši jednotlivé skupiny uživatelů. U každé skupiny pak uvádím diagram případů užití a v textu probírám specifika a jednotlivé případy užití skupiny. Vzhledem jasnému a popisnému rozdělení jednotlivých případů užití neuvádím jejich detaily. K popisu detailů z velké části slouží především návrh uživatelského rozhraní (kapitola 2.5.2).

Závodník

Závodníci přijdou do kontaktu s terminálem pouze minimálně. Jedinou operací, kterou budou na terminálu provádět samostatně je přečtení dat z čipu SI a odebrání výtisku se svými výsledky. Ve všech ostatních případech, bude mezi závodníkem a terminálem operátor, který bude terminál obsluhovat. Závodníka tedy není možné zatěžovat jakoukoliv složitější manipulací. Jeho případy užití jsou tedy pouze 2 vzájemně se doplňující. V prvním kroku závodník najde a zobrazí detail své přihlášky a v druhém kroku vyčte data z čipu SI. Vzhledem k tomu, že SI čip poskytuje možnost jasné identifikace závodníka, je možné ve výsledku tyto 2 případy sloučit do jedné operace – zasunutí čipu do kontrolní jednotky terminálu. Terminál se postará o zbytek. Tedy automaticky vyčte data, zobrazí detail vyčtených dat na displeji, ohlásí vyčtení dat zvukovým znamením a vytiskne výsledek na tiskárně.

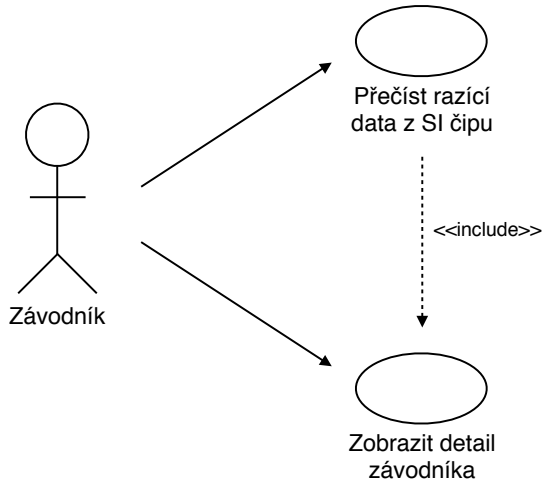


Schéma 2.3 – Diagram případů užití – závodník

Operátor prezentace

Operátoři prezence mají za úkol odbavit závodníka při příchodu na shromaždiště události a obsloužit případně požadavky na organizační změny. Jedná se například o zapsání nového závodníka včetně výběru kategorie, trati a přiřazení startovacího času, změnu osobních údajů závodníka či změnu závodníkova čipu SI, startovního času nebo kategorie a odstranění závodníka, který se nedostaví.

Společným prvkem většiny operací je vyhledání a zobrazení detailu přihlášky závodníka. Tento případ lze provést standardním výběrem závodníka ze seznamu, ale také vložením čipu SI do kontrolní jednotky terminálu. Stejně tak lze provést zápis nového čísla SI čipu závodníka – tedy přímým zadáním nebo vložením čipu do terminálu.

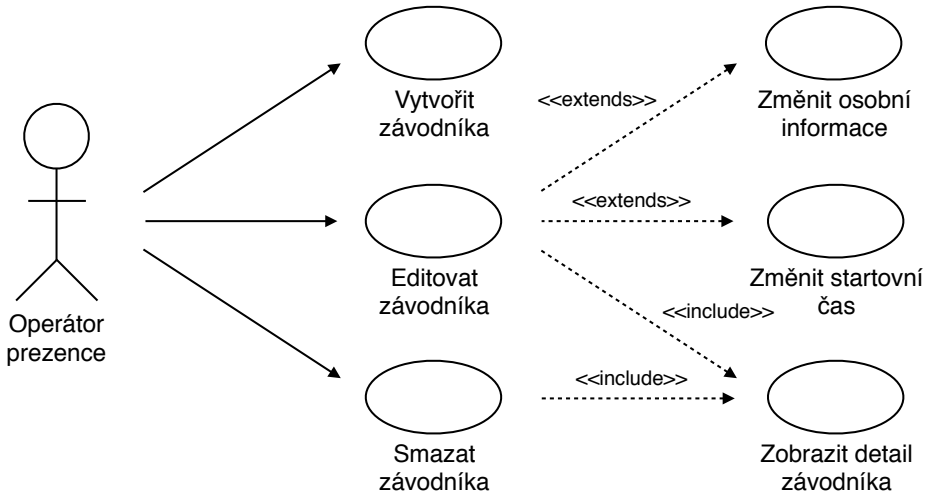


Schéma 2.4 – Diagram případů užití – operátor prezence

Operátor startu

Na místě startu události slouží aplikace především k označování průchodů startem jednotlivých závodníků a kontrole jejich startovních časů. Obsluha terminálu po začátku příslušné startovní minuty postupně nechá vkládat startující závodníky své SI čipy do terminálu, který zkontroluje jejich startovní čas a zapíše jejich průchod startem do datového modelu aplikace.

Na místě startu mohou operátoři řešit některé nestandardní situace, jako je výměna závodníkova SI čipu za nový v případě poruchy starého či změna přiřazeného startovního času závodníka. Opět se zde vyskytuje společný případ, kterým je vyhledání a zobrazení detailu závodníka, které je možné provést výběrem ze seznamu nebo vložení závodníkova SI čipu do kontrolní jednotky.

Operátoři startu kladou z uživatelů nejvyšší důraz na mobilitu a ergonomii celého terminálu.

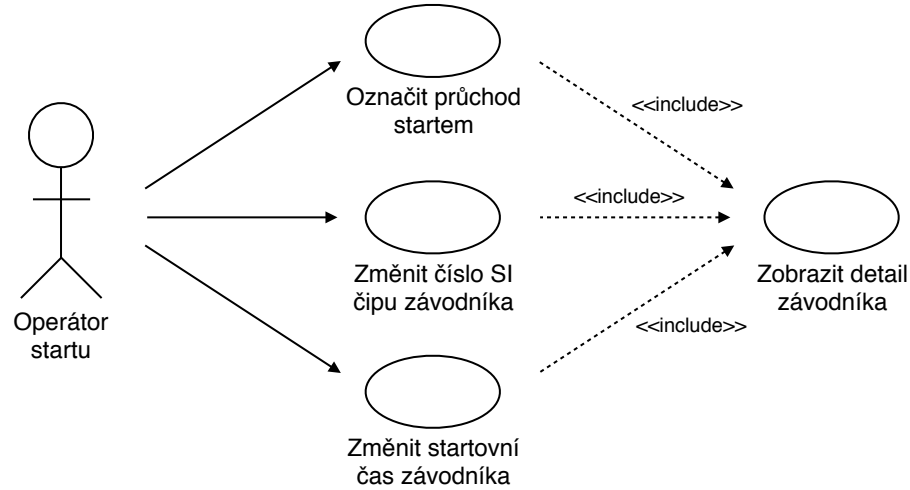


Schéma 2.5 – Diagram případů užití – operátor startu

Operátor časomíry

Operátoři časomíry sídlí většinou v centru události. Díky integrovanému terminálu se jejich činnost omezila pouze na řešení požadavků na ruční editaci dat či stavu závodníků. K takovým situacím dochází například při poruše kontrolních jednotek na trati závodu či diskvalifikaci závodníka z důvodu porušení pravidel.

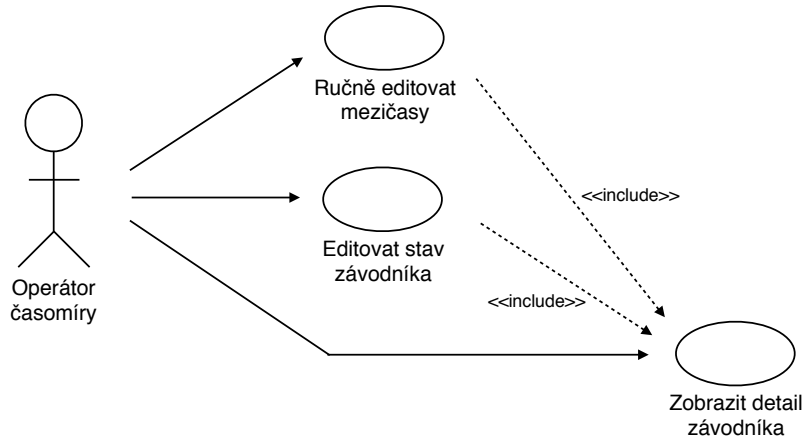


Schéma 2.6 – Diagram případů užití – operátor časomíry

Administrátor události

Role administrátora události. Je důležitá především před začátkem události. Administrátor musí nakonfigurovat terminál pro následující událost. Jeho úkolem je tedy importovat data události z jiných systémů, či připojit terminál k jinému terminálu. Aplikace může naráz obsahovat data více událostí, veškeré operace se ale provádějí vždy nad jednou událostí, administrátor tedy nastavuje aktivní událost. Administrátor dále exportuje výsledná data do jiných systémů a má možnost odstranit data konkrétní události z aplikace.

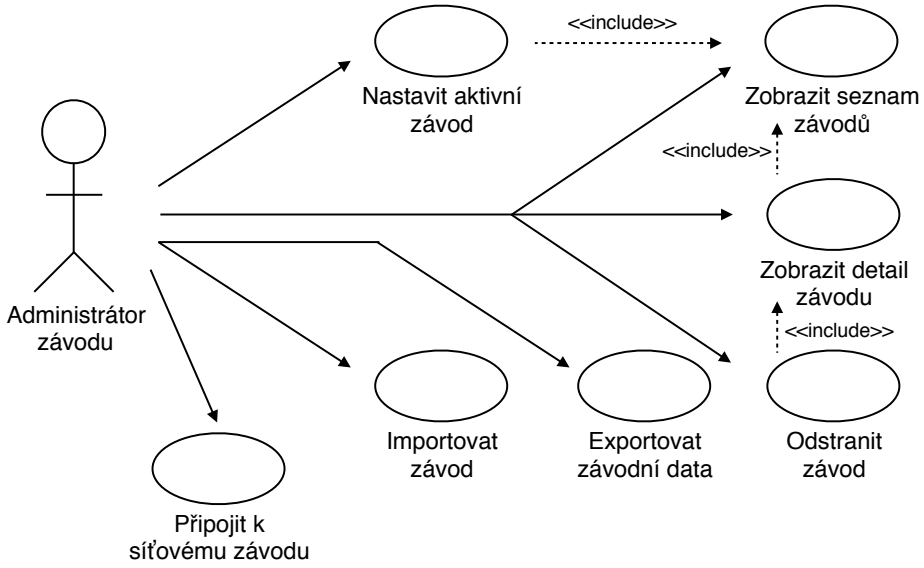


Schéma 2.7 – Diagram případů užití – administrátor aplikace

2.5.2 Návrh uživatelského rozhraní

Vzhledem různorodé skupině budoucích uživatelů je uživatelské rozhraní rozděleno tak, aby každý uživatel měl co nejrychlejší přístup k funkcím které vyžaduje. Uživatelské rozhraní aplikace tedy obsahuje záložky, jejichž obsah přibližně odpovídá potřebám jednotlivých skupin. Rozdělení do záložek je řešeno pomocí UITabBar a UITabBarController komponent.

Aplikace po svém opětovném spuštění zůstává ve stejném stavu jako před ukončením. Aplikace se automaticky spustí při zasunutí SI čipu do kontrolní jednotky terminálu.

V následujících podkapitolách popisuji postupně všechny záložky a k většině z nich uvádím i drátěné modely obrazovek. Zobrazeny nejsou vždy všechny záložky. Některé se skrývají pod poslední záložkou more, která po doteku zobrazí seznam skrytých záložek. Uživatel může pořadí záložek změnit. Jedná se o standardní chování komponenty UITabBarController.

Většina prvků uživatelského rozhraní jsou standardními prvky z knihovny UIKit. Tyto prvky je možné nalézt ve většině aplikací pro iOS a logika jejich ovládání je tedy pro uživatele velmi intuitivní.

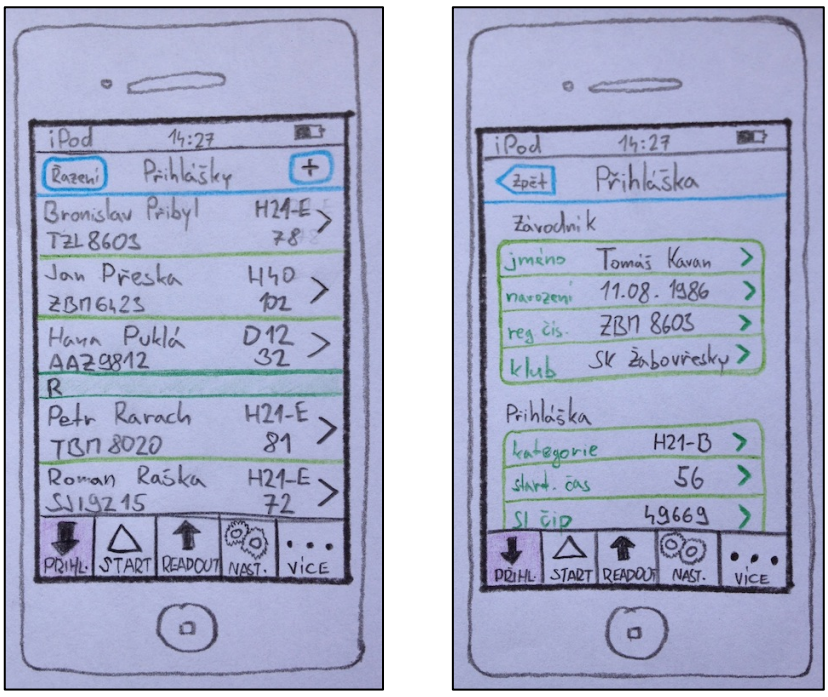
Přihlášky

Záložka přihlášky slouží uživatelům ze skupiny operátor časomíry a operátor prezence ke správě dat o závodnících. Obrazovky v této záložce dovolují přidávat nové závodníky, editovat údaje stávajících, korigovat data časomíry doběhnuvších či upravovat celkový stav výsledku závodníka.

Záložka začíná na obrazovce „seznam přihlášek“, která je zobrazena na obrázku 2.2 vlevo. Obrazovka poskytuje přehled všech přihlášených závodníků v závodě. Seznam je možné filtrovat vyhledávacím polem nebo řadit dle příjmení, kategorie, startovního času či klubové příslušnosti. Každý závodník je v seznamu reprezentován těmi nejzákladnějšími informacemi, tedy svým jménem, registračním číslem, kategorií a startovním časem. Seznam je vytvořen standardním prvkem UITableView a UITableViewCell.

V pravé horní části seznamu je, umístěno tlačítko pro přidání nového závodníka, které uživateli zobrazí prázdný detail přihlášky (obrázek 2.2, pravá obrazovka). Detail přihlášky v takovém případě neobsahuje tlačítko zpět, ale tlačítka uložit a zrušit. Bez vyplnění polí jméno, narození, registrační číslo, klubu, kategorie, tratě a startovního času není možné novou přihlášku uložit. V případě pokusu přihlásit závodníka, který má s jiným závodníkem shodné registrační číslo, startovní číslo, či číslo SI čipu aplikace zobrazí chybovou informaci a nedovolí přihlášku uložit.

Při doteku řádku konkrétního závodníka, či vložení příslušného SI čipu do terminálu se zobrazí detail jeho přihlášky. Jednotlivé údaje je možné editovat dotekem příslušného řádku. Tímto se zobrazí další obrazovka pro editaci konkrétní hodnoty podobně, jak je tomu na obrázku 2.4 (pravá obrazovka), pouze s jiným vstupním polem a typem klávesnice. Editační obrazovka překrývá i spodní řádek záložek, aby nebylo možné odejít z editační obrazovky jinak, než potvrzením, či zrušením změn.

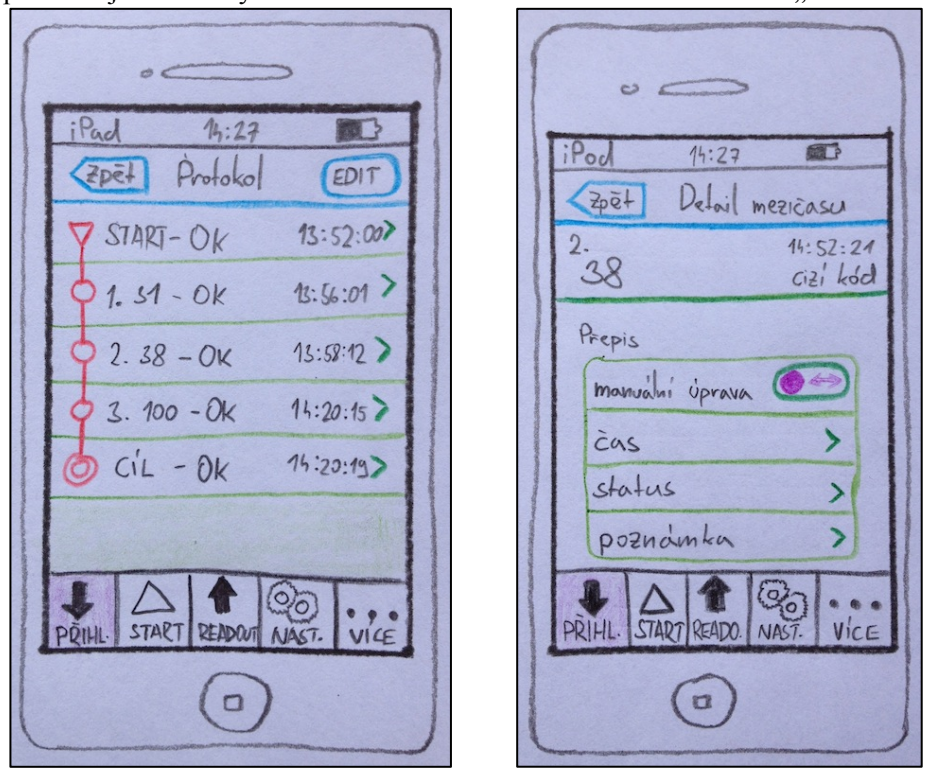


Obrázek 2.2 – Obrazovky seznamu a detailu přihlášek

Detail přihlášky obsahuje řádek „protokol ražení“. Dotek tohoto řádku zobrazí obrazovku protokolu ražení (obrázek 2.3 – levá obrazovka). Protokol ražení poskytuje informace o datech přečtených z SI čipu závodníka po doběhnutí do cíle. Pokud závodník ještě po doběhu čip v terminálu nepřečetl, je zobrazený seznam kontrol jeho tratě s kontrolami ve stavu „missing“ (viz dále). Z důvodu možnosti poruchy razících jednotek na trati či jiných nepředvídatelných situací musí mít obsluha terminálu možnost editovat razící data manuálně.

Jak závodníci postupně čtou své SI čipy po doběhnutí do cíle, tvoří se log výčtů ražení. Závodníkovi je možné vybrat jakýkoliv výčet ražení ze seznamu doposud provedených výčtů. Toto se provádí tlačítkem „vybrat výčet“, které zobrazí obrazovku seznamu výčtů (viz dále – záložka ražení). Zde je možné závodníkovi přiřadit jakýkoliv výčet. Při přiřazení nového výčtu nahradí nová série ražení tu původní a pokud existují manuální úpravy původního ražení, aplikace se zeptá, zda je chce uživatel ponechat či odstranit. Pokud přiřazený výčet neobsahuje všechny předepsané kontroly tratě závodníka, zobrazí se tyto kontroly v seznamu také, ovšem se stavem „missing“.

Celý protokol je možné vytisknout tiskárnou terminálu dotekem tlačítka „tisk“.



Obrázek 2.3 – Obrazovky protokolu razících dat a detailu ražení

V seznamu není možné přidávat, měnit pořadí či odstraňovat jednotlivé položky. Dotekem jednotlivých řádků seznamu, se ale zobrazí obrazovka pro manuální editaci položky (obrázek 2.3 – pravá obrazovka). Pro přepis času či stavu položky je třeba aktivovat přepínač „manuální úprava“ a vepsat čas ražení, či změnit stav. Razicí záznam se může nacházet v jednom z těchto stavů:

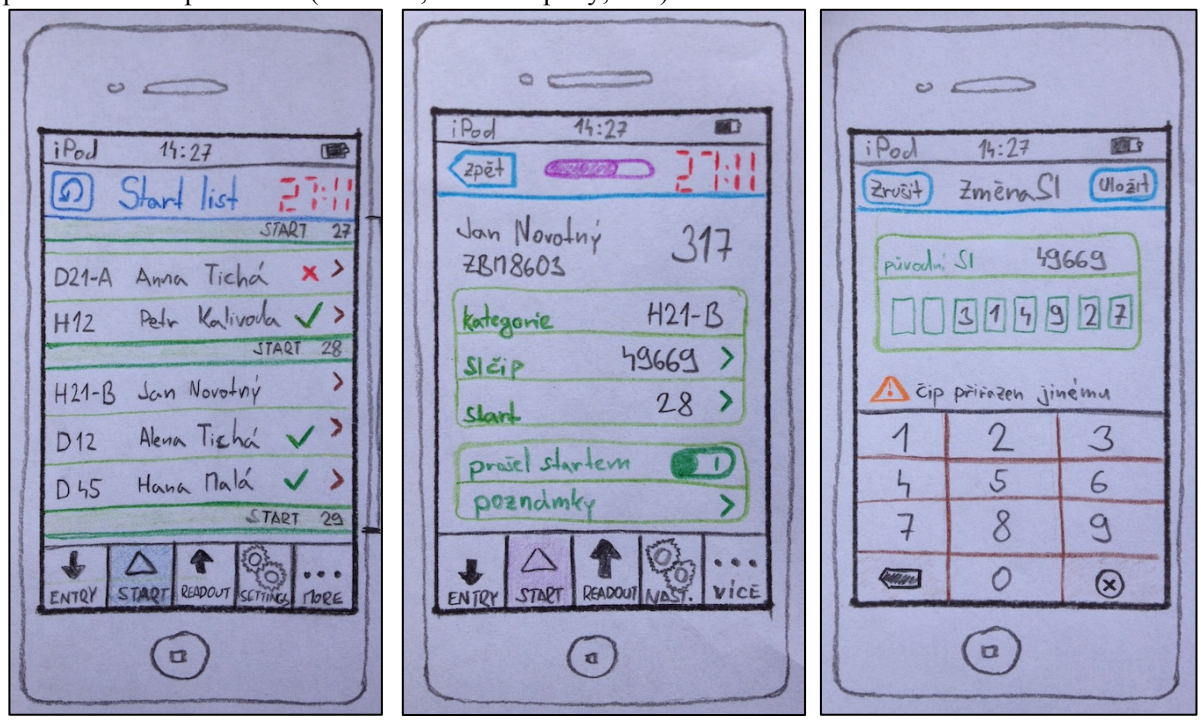
- OK – ražení je v pořádku,
- OKnolap – ražení je v pořádku, ale bez mezičasu,
- wrong – chybná kontrola,
- missing – chybějící předepsaná kontrola.

Aplikace nebrání vepsání dekrementálních časů ražení – tedy předcházejících času ražení předchozího záznamu. Toto je v kompetenci operátora časomíry.

Start

Záložka start slouží operátorům startu především k předstartovní kontrole startujících závodníků. Před vstupem do prvního startovního koridoru provádí operátor startu porovnání startovního času závodníka s aktuálním časem v prvním koridoru podle čísla SI čipu závodníka. Dle pravidel OSP je pořadatel povinen závodníkovi poskytnout na startu náhradní čip, pokud jeho původní není funkční. Operátor startu tedy musí mít možnost závodníkovi přiřadit čip nový. Zřídka je také třeba změnit startovní čas závodníka. Data o závodnících prošlých startem je třeba přenést do centra závodu.

Základní obrazovkou záložky start je startovní listina (viz obrázek 2.4 – levá obrazovka), neboli seznam závodníků seřazený a seskupený podle startovního času. V levé části navigační lišty je tlačítko pro vycentrování startovní listiny na aktuální startovní minutě. V pravé části navigační lišty jsou zobrazeny startovní hodiny prvního koridoru (první koridor je obvykle oproti startu posunut o 3 minuty). U každého závodníka je v případě jeho průchodu startem tato skutečnost označena symbolem. Startovní listina se na začátku každé minuty automaticky posune na příslušnou minutu. Celá záložka start, je navržena tak, aby se, kromě speciálních případů, jako je změna čipu, nebylo nutné prsty dotýkat obrazovky. A to z důvodů možné špatné manipulace v případě zhoršených povětrnostních podmínek (rukavice, necitlivé prsty, atd.).



Obrázek 2.4 – Obrazovka startovní listiny, detailu startujícího závodníka a změny SI čipu

Detail závodníka (viz obrázek 2.4 – uprostřed) se zobrazí vložením jeho čipu do kontrolní jednotky terminálu nebo dotekem příslušného řádku ve startovní listině. Při vložení čipu a dojde k automatickému označení průchodu startem. V případě absence doteku obrazovky se detail závodníka po 3 sekundách automaticky skryje. Odpočet do skrytí detailu závodníka vyjadřuje snižující se indikátor pokroku v prostřední části navigační lišty.

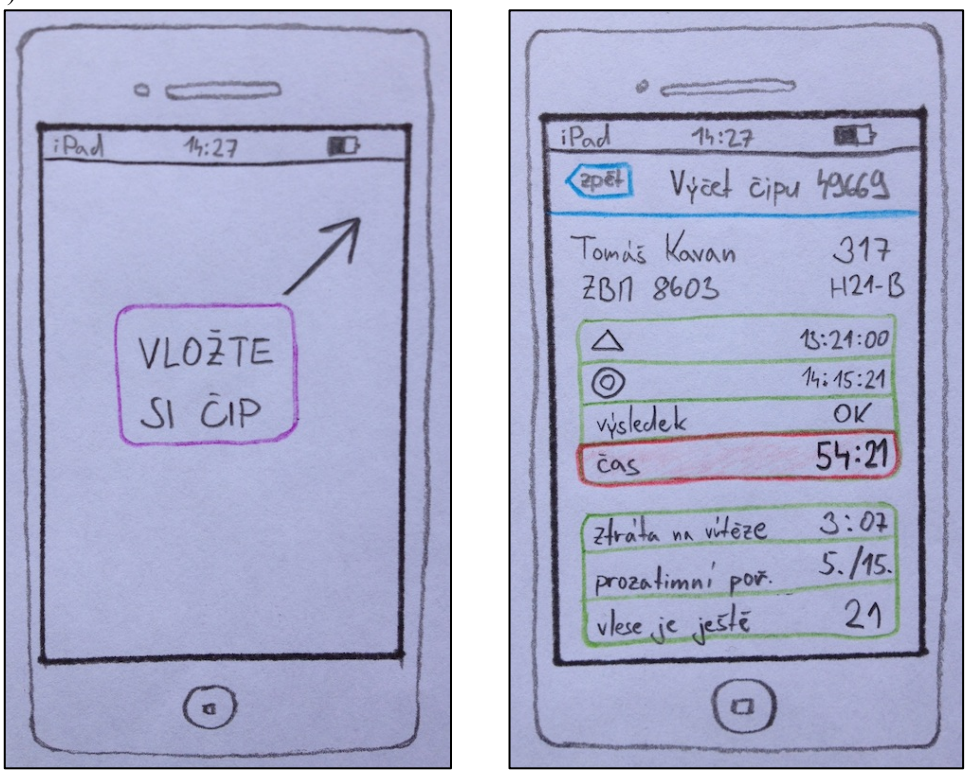
Obrazovka poskytuje nejdůležitější předstartovní informace o závodníkovi (jméno, startovní čas, startovní číslo, kategorii, trať, číslo SI čipu a registrační číslo). Číslo SI čipu a startovní čas může operátor startu měnit dotekem příslušného řádku s údajem. Operátor startu také může k závodníkovi vepsat poznámku.

Změna SI čipu, startovního času, či vepsání poznámky se provádí na speciální obrazovce, obsahující jen měněný údaj. V případě změny SI čipu (viz obrázek 2.4 – pravá část) je možné zadat

číslo čipu vložením nového čipu do kontrolní jednotky SI. Aplikace při zadávání SI čipu automaticky hlídá duplicity a neumožní zvolit stejné číslo čipu pro 2 závodníky.

Vyčítání čipů

Jedinou záložku, se kterou přijdou do styku uživatelé ze skupiny závodník je výčet čipů. Základní obrazovkou záložky je výzva pro vložení čipu do kontrolní jednotky terminálu (viz obrázek 2.5 – levá obrazovka).



Obrázek 2.5 – Obrazovka výzvy k vložení SI čipu a detailu zobrazeného při tisku protokolu

Po vložení čipu závodníkem dojde k automatickému vyčtení dat z čipu, zobrazení výsledku závodníka (viz obrázek 2.5 – pravá obrazovka) a vytištění protokolu tiskárnou terminálu. Obrazovka s výsledkem závodníka se automaticky po několika sekundách skryje. Závodník tedy nemusí s terminálem interagovat žádným dotekem a proces výčtu čipu je pro něj známý již ze současnosti.

V případě vložení SI čipu, který není v systému znám, zobrazí aplikace místo výsledku informaci o nutnosti přivolání obsluhy. Výčet čipu se ale i tak provede a zapíše do logu (viz záložka ražení).

Kluby

Záložka kluby slouží pro správu klubů registrujících závodníky přihlášené na závod. Jedná se o doplňkovou funkcionalitu, protože seznam klubů musí být importován do aplikace společně s přihláškami. Nový klub bude chtít operátor prezence založit jen v případě dohlášení závodníka v místě závodu.

Základní obrazovkou záložky kluby je seznam klubů, seřazený abecedně, podle zkratky klubu. V seznamu je možné vyhledávat. Dotekem tlačítka „přidat“, se zobrazí prázdná obrazovka detailu klubu. Při přidávání nového klubu je nutné vyplnit alespoň jeho zkratku. V opačném případě nedovolí aplikace klub přidat.

Dotekem řádku klubu se zobrazí obrazovka s editovatelným detailem klubu a tlačítkem pro jeho odstranění. Odstranění aplikace neumožní provést, pokud je ke klubu v závodu přiřazen alespoň

jeden závodník. Detail klubu je editační obrazovkou, je proto vždy zobrazena s tlačítky „uložit“ a „zrušit“ v navigační liště, vysunutou klávesnicí překrývající záložky a fokusem na první textové pole. Klub je definován pouze zkratkou a jménem.

Obrazovka seznam klubů je využita v detailu přihlášky při editaci klubu závodníka. V takovém případě je zobrazena jako modální view controller, který neumožňuje přidání nového klubu a dotekem řádku klubu se tento klub přiřadí editovanému závodníkovi.

Ražení

Záložka ražení slouží pro zobrazení logu vyčtených čipů. Základní obrazovkou je seznam výčtů čipů SI. Defaultně je řazen dle času vyčtení, ale tlačítkem „Řazení“ je možné zobrazit modální obrazovku dovolující zvolit mezi řazením dle čísla SI čipu a časem výčtu. Záznamy o vyčtení není možné jakkoliv editovat či odstranit. Každý záznam, který je přiřazen ke konkrétnímu závodníkovi – typicky se jedná o většinu záznamů – má u svého řádku tuto informaci uvedenou.

Při doteku řádku se zobrazí obrazovka s detailem ražení. V detailu jsou uvedeny všechny data vyčtená z čipu včetně času vynulování a kontroly čipu, záznamů jednotlivých průchodů kontrolními stanovišti a cílovým časem. Data opět není možné editovat.

Export

Z aplikace je možné exportovat veškerá závodní data ve formátu IOF XML Data Standard. Jedná se o data typu seznam událostí, seznam přihlášek, výsledkovou listinu a data běhů (viz kapitola 1.4.1). V záložce export je možné vybrat vždy jeden z těchto seznamů, či zvolit možnost všechna data. Exportují se vždy jen data k aktivní události.

V dalším kroku aplikace zobrazí možnosti cíle exportu. Vzhledem k systému správy souborů v systému iOS, je tento krok mírně problémový, data je možné pouze exportovat do souboru, který je možné zaslat nějaké konkrétní aplikaci podporující soubory XML. Základní možností je tedy pouze odeslat exportovaná data jako přílohu emailu, nebo vytisknout pomocí technologie AirPrint[30].

Připojení

Záložka připojení slouží administrátorovi aplikace k připojení terminálu ke vzdálené události či k zobrazení připojených terminálů. Jeden terminál tedy musí být hlavní (hostitel) a připojené terminály využívají jeho datový model jako vlastní (klienti).

Základní obrazovka zobrazuje stav připojení. Tedy, je-li terminál samostatný, připojen k jinému terminálu nebo očekává připojení jiných terminálů ke svému datovému modelu s případným počtem připojených terminálů. Dále základní obrazovka poskytuje trojstavový přepínač pro volbu módu (samostatný, klient, hostitel).

Při zvolení samostatného módu se aplikace dotáže, zda-li se chce operátor opravdu odpojit od hostitelského terminálu či odpojit klientské terminály. Pokud operátor požadavek potvrdí, přepne se aplikace do samostatného módu. V případě, že před tím byla v klientském módu, odstraní lokální cache závodních dat a vybere jako aktivní závod jeden z lokálních.

Po zvolení klientského módu se zobrazí seznam dostupných závodů (hostitelských terminálů) k připojení. Po vybrání jednoho ze závodů se zobrazí vstupní pole pro zadání uživatelského jména a hesla. Po úspěšném procesu autentizace terminál stáhne z hostitele datový model a použije jej jako cache.

Po zvolení hostitelského módu se zobrazí vstupní pole pro zadání uživatelského jména a hesla. Po potvrzení je terminál nakonfigurován jako hostitel a jeho aktuální závod zvolen jako síťový.

Rozpoznávání hostitelských terminálů je implementováno pomocí technologie Bonjour[31].

[30] Technologie dovolující přímý tisk ze zařízení iOS bez speciálních tiskových ovladačů
[31] Technologie nulové konfigurace pro propagaci síťových služeb v místní síti.

Události

Záložka události slouží administrátorovi pro správu datových modelů závodních událostí uložených v terminálu, či pro import dat.

Základní obrazovka zobrazuje seznam uložených událostí a tlačítko pro import či založení nové události. U každé události je uveden její název a datum konání. U aktivní události je navíc symbol upozorňující na tuto skutečnost.

Při doteku řádku z událostí se zobrazí detailní informace o události. Detail události umožňuje nahrání uložených importních souborů. Dále je zde volba označení události jako aktivní a volba smazání události.

V případě, že je aktivní hostitelský či klientský režim, není možné nastavit jinou aktivní událost a ani aktivní událost smazat. Aplikace na tuto skutečnost uživatele vždy upozorní.

Import dat

Import dat do aplikace je prozatím, vzhledem k organizaci souborů v systému iOS, nejméně ergonomickou částí aplikace. Nejjednodušší cestou jak, nahrát do aplikace importní XML soubory, je zaslat jako přílohu emailu a tu potom v telefonu uložit do aplikace. V budoucnu mám v plánu import dat výrazně zjednodušit například implementací veřejně dostupné webové služby.

2.5.3 Datový model

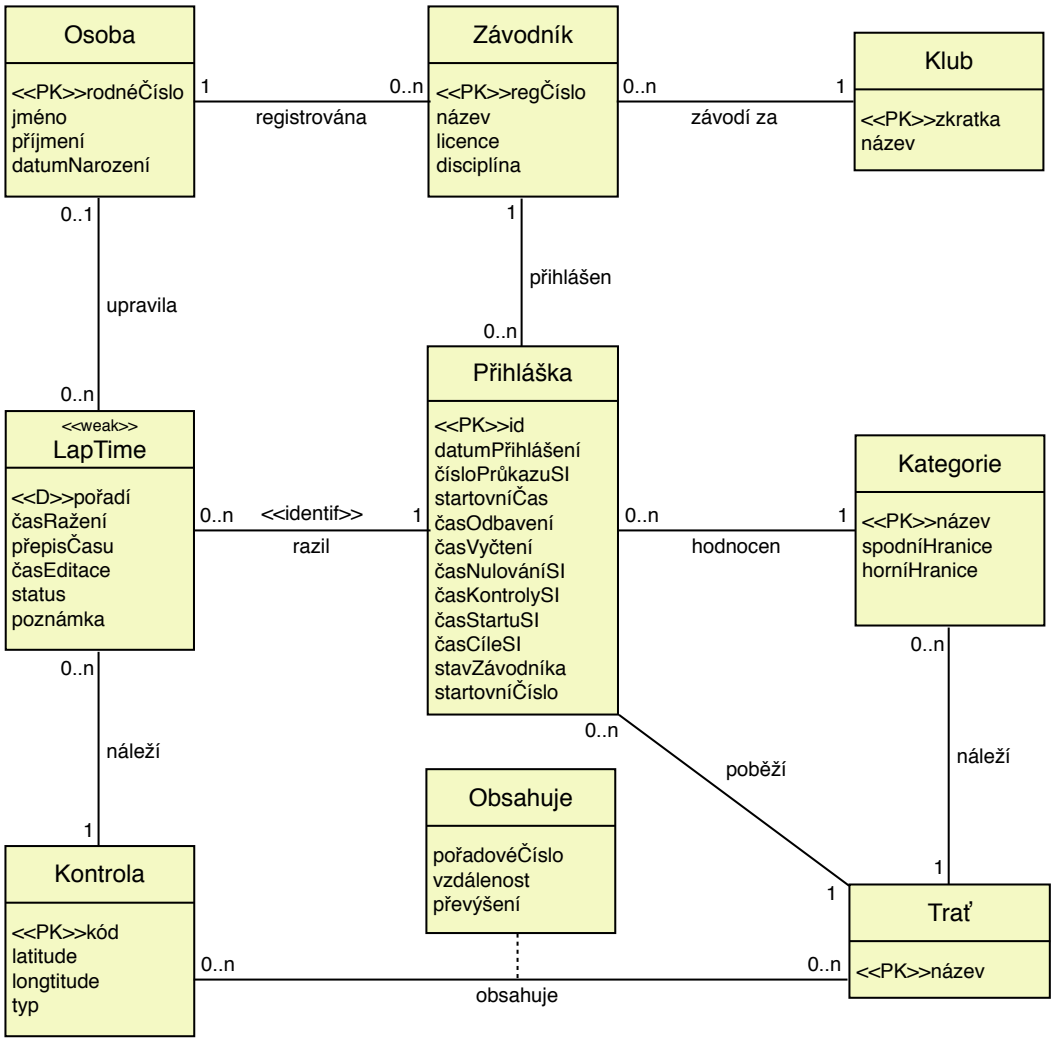


Schéma 2.8 – ER diagram aplikace EasyEvent – závod

Datový model aplikace popisuje datové entity a vztahy mezi nimi (relace), nad kterými aplikace EasyEvent staví své uživatelské rozhraní. Protože aplikace pracuje vždy jen nad jedním aktivním závodem, není nutné do návrhu datového modelu zapojovat entitu události. Datový model události vyjádřený ER diagramem (viz schéma 2.8) tedy neobsahuje žádné entity pro identifikaci záznamů konkrétní události. V reálné implementaci je tedy tyto záznamy nutné fyzicky rozdělit – například použít jinou databázi.

Za hlavní datovou entitu je možné považovat entitu Přihláška, která závodníkovi přiřazuje Kategorii, Trať či seznam výsledků (LapTime). Přihláška také shromažďuje všechny informace o konkrétním závodníkovi relevantním speciálně pro daný závod. Jedná se například o číslo jeho SI průkazu, startovní čas, či časy získané v průběhu závodu (nulování čipu, kontrola čipu, čas cíle, atd.). Entita Přihláška také uchovává celkový stav závodníka. Pokud je tedy závodník z jakéhokoliv důvodu diskvalifikován, je mu nastaven právě příslušný stav závodníka.

Data získaná vyčítáním závodních typů jsou odděleny do speciální ER diagramu (viz schéma 2.9), protože s daty o závodě nemají mnoho společného. Jeden výčet průkazů tedy obsahuje informace o kontrolní jednotce, vyčteném průkazu a všech razících dat, uložených v průkazu. Při přiřazování výčtu konkrétnímu závodníkovi se výčet se závodníkem páruje podle čísla SI čipu závodníka a razící data se vytvoří jako instance entity LapTime, zároveň se některá obecná data výčtu propíší do příslušné závodníkovi přihlášky. Mezi entitou Ražení a LapTime ani VýčetPrůkazu a Přihláška záměrně není žádná pevná vazba (relaci).

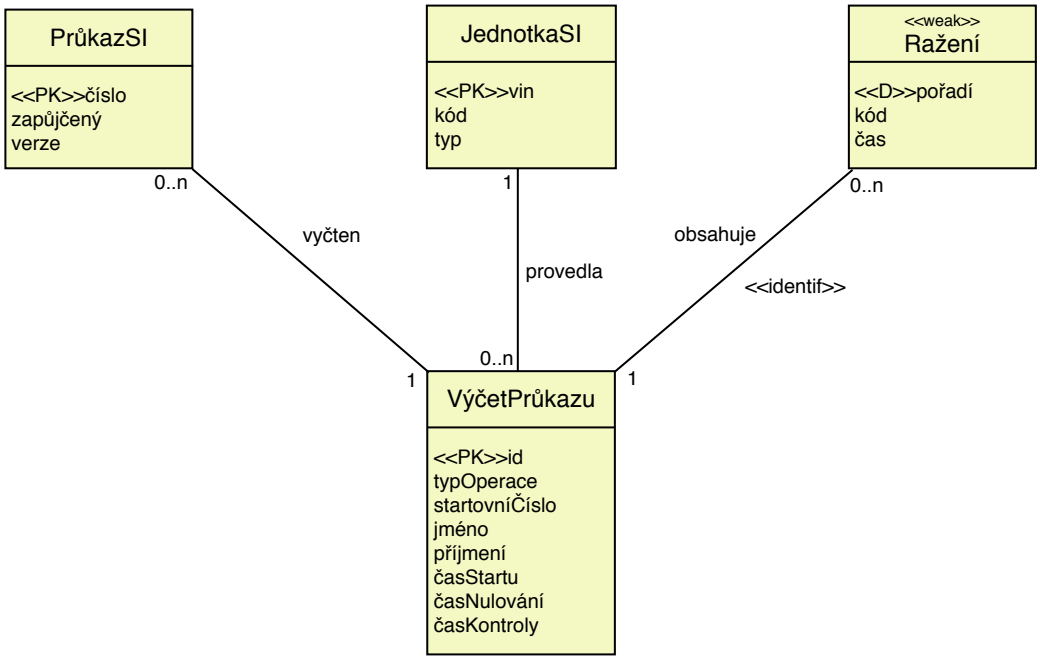


Schéma 2.9 – ER diagram aplikace EasyEvent – databáze ražení

Entita kontrola reprezentuje fyzickou kontrolu v terénu, kterou je možno označit do SI čipu závodníka. Kontrola je jednoznačně identifikována svým kódem. Typ kontroly určuje, zda-li se jedná o běžnou, startovní či cílovou kontrolu. Kontrola má vazbu na tratě, které kontrolou procházejí. Tato relace má několik parametrů. Protože je v rámci tratě pořadí kontrol předepsáno, je parametrem pořadí kontroly na trati, dále pak vzdálenost a převýšení od poslední kontroly. Dle těchto parametrů se počítá celková délka a převýšení tratě (převýšení se uvádí vždy jen kladné, tedy stoupání).

Kategorie může mít přiřazeno více tratí, toho využívají některé typy závodů – například s hromadným startem – kde závodníci jedné kategorie mají různou trať, ale hodnocení jsou společně. Jedna trať také může být přiřazena více kategorií. Často se například spojují kategorie juniorských a

seniorských kategorií. Přihláška pro správné vyhodnocení závodu tedy potřebuje nejen relaci na konkrétní trať, ale i kategorii.

Relace mezi osobou a LapTime slouží pro identifikaci operátora, který manuálně upravil razící data.

2.5.4 Logická architektura aplikace

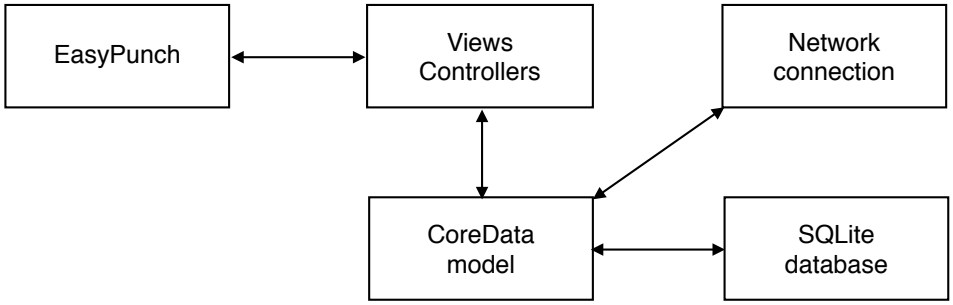


Schéma 2.10 – Logická struktura aplikace EasyEvent

Aplikaci EasyEvent je možné logicky rozdělit do několika částí (viz schéma 2.10). Část uživatelského rozhraní nazvaná „Views & Controllers“ slouží k provozu běhové smyčky a interakci s uživatelem, popsané v kapitole 2.5.2. Součástí této části jsou i třídy pro inicializaci aplikace a vytvoření objektového grafu.

Datový model aplikace nazvaný „Core Data model“ je druhou logickou částí aplikace, která poskytuje oddělovací vrstvu mezi fyzické úložiště dat či sítové připojení a část uživatelského rozhraní. Datový model neslouží pouze jako implementace ER diagramů popsaných v kapitole 2.5.3, ale i pro obecnou komunikaci s externě připojenými terminály.

Instance EasyPunch je Objective-C obalem nad knihovnou EasyPunch komunikující s připojeným mikrokontrolérem, tiskárnou a základovou stanicí terminálu. Část je plně řízena částí uživatelského rozhraní, protože lze v podstatě považovat za HID (Human Interface Device).

Konkrétní implementační detaily aplikace EasyEvent popisuji dále v kapitole 3.2, ale především v samotných zdrojových kódech aplikace přiložených na paměťové kartě k této bakalářské práci.

3 Realizace

V kapitole popisuji implementační a vývojové detaily z tvorby prototypového terminálu. Nejdříve proberu použité postupy, materiály a technologie při vývoji hardwarové části terminálu (kapitola 3.1). Později se zmíním o vývoji softwarového vybavení terminálu, tedy aplikace EasyEvent, knihovny EasyPunch a programu mikrokontroléru (kapitola 3.2). Speciálně se zabývám vývojovými nástroji a postupem překladu a instalace vytvořeného software. V další části kapitoly navrhuji uživatelské testy prototypu terminálu, včetně jejich účelu, odhadované zpětné vazby a návrhu metrik (kapitola 3.3). V poslední kapitole zmiňuji plánované reálné nasazení prototypu (kapitola 3.4).

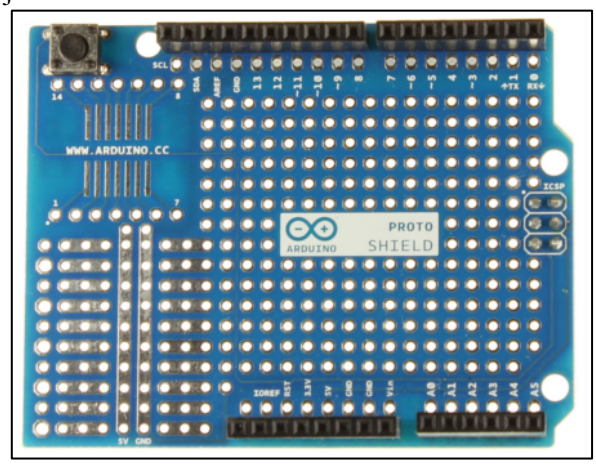
3.1 Vývoj prototypu terminálu

Vývoj hardwarové části prototypu terminálu jsem začal testy komunikace kontrolní jednotky SPORTIdent s iOS zařízením přes Redpark TTL kabel, vývojového kitu Arduino Uno s tiskárnou Adafruit Mini Thermal Printer a kitu Arduino Uno s iOS zařízením. V testech jsem zprovoznil komunikaci mezi všemi jednotkami, zafixoval si teoreticky nastudované znalosti o protokolu SPORTIdent a vývoji pro vývojový kit Arduino. Zároveň jsem měl v praxi možnost vyzkoušet komunikaci iOS zařízení přes Redpark TTL kabel.

3.1.1 Vývoj a výroba podpůrného obvodu mikrokontroléru

Stavbu podpůrného obvodu mikrokontroléru a propojení všech komponent terminálu jsem provedl v několika krocích. Prvním krokem bylo zapojit celý obvod na nepájivém kontaktním poli pro testování funkčnosti. Samotné zapojení není příliš složité a je zobrazenu v příloze B.

Následovala volba vhodné desky pro plošný spoj. Návrh a výroba vlastní desky se pro účely prototypu jevila jako časově příliš náročná a neekonomická. Vzhledem k otevřenosti a modularitě vývojových kitů Arduino jsem zvolil Arduino Proto Shield



Obrázek 3.1 – Arduino Proto Shield (Zdroj: http://arduino.cc/en/Main/ArduinoProtoShield)

Arduino Proto Shield je předleptaná předvrtaná vývojová deska plošných spojů rozměry a vývody odpovídající vývojovým kitům Arduino. Desku je možné přímo položit na Arduino a tím propojit všechny kontakty. Realizace navrženého podpůrného obvodu na této desce byla jednoduchá a k výrobě postačila pájecí stanice, modelářský svěrák, dutinkový cín s kalafunou, čistící roztok na kalafunu, miniaturní štípací kleště, odsávačka přebytečného cínu a digitální multimetr.

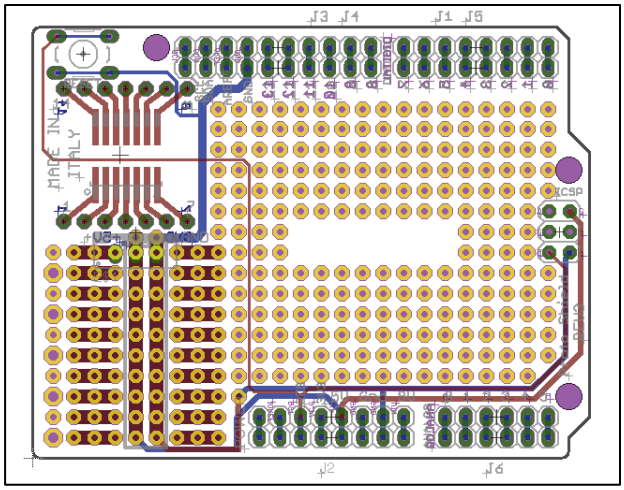


Schéma 3.1 – Arduino Proto Shield výkres DPS (Zdroj: http://arduino.cc/en/Main/ArduinoProtoShield)

3.1.2 Výroba obalu prototypového terminálu

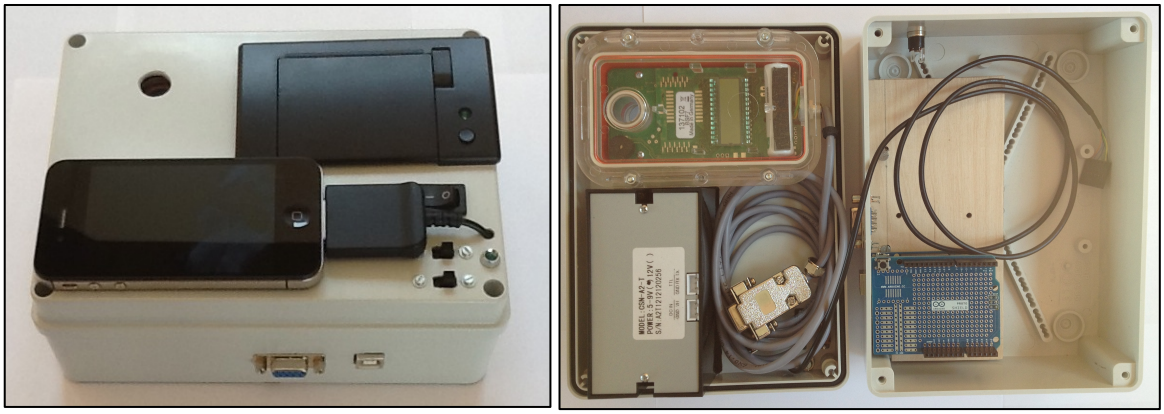
Požadavky na obal prototypového terminál jsem oproti obalu budoucí produkční verze terminálu výrazně zredukoval. Netrval jsem tolik na ergonomii a na krytí IP 67. To především z důvodu omezené doby použití prototypového terminálu a nutnosti externího zdroje.

Pro prototyp terminálu jsem zvolil univerzální plastovou konstrukční krabici U-AS35 o rozměrech 190x140x70 cm s krytím dle normy IEC60529 IP56. Tedy odolnou proti prachu a proti vlnobití.

Před úpravou krabičky jsem připravil náčrt budoucích konstrukčních otvorů a rozmístění i přichycení vnitřních prvků v krabici. Samotnou úpravu krabičky jsem provedl pomocí modelářské vrtačky a brusky. Vytvořené konstrukční otvory jsem začistil a vyhladil pilníkem a jemným smirkovým papírem. Na okraje otvorů, které i po instalaci komponent zůstaly viditelné jsem nanesl bezbarvý lak.

Kvůli tvorbě konstrukčních otvorů ztratila krabička své původní krytí. Z toho důvodu bylo nutné všechny komponenty vyplňující konstrukční otvory a jejich spojení s krabičkou zajistit tímto krytím. Konektory jsou opatřeny gumovými průchodkami a veškeré prvky jsem z vnitřní strany zalepil pomocí tavící pistole.

Slabým článkem prototypu je kabel Redpark a iOS zařízení, které jsou umístěny vně krabice a nemají žádné dodatečné krytí. Obsluha prototypu terminálu se musí k těmto komponentám chovat příslušně šetrně.



Obrázek 3.2 – Hotový prototyp terminálu (zkompletovaný a otevřený bez většiny kabeláže)

3.2 Implementace softwarové části

Vývoj softwarového vybavení terminálu postupoval společně s vývojem hardware. Nejdříve jsem programoval testovací programy pro osvojení sériové komunikace na iOS a Arduinu, dále jsem se seznámil s Redpark SDK a otestoval komunikaci s jednotkou SPORTIdent.

S takto nabytými zkušenostmi jsem se začal věnovat vývoji knihovny EasyPunch. Knihovnu jsem navrhoval a vyvíjel dvakrát, protože první návrh byl v praxi absolutně nepoužitelný. Výslednou knihovnu jsem otestoval s jednotkou SPORTIdent a v komunikaci mezi dvěma samostatnými instancemi knihovny. Po testech jsem mohl připravit platformní porty knihovny pro iOS a Arduino. Otestováním komunikace byla práce na knihovně hotova.

Dalším krokem ve vývoji byla implementace aplikace EasyEvent. Uživatelské rozhraní aplikace jsem implementoval pomocí standardních komponent dodávaných společně iOS SDK. Nejzajímavější částí implementace byl vývoj datového modelu pomocí frameworku Core Data, který jsem použil i pro implementaci větší části komunikace se vzdálenými terminály.

3.2.1 Vývojové nástroje

Pro veškeré softwarové práce jsem použil integrované vývojové prostředí Xcode dodávané firmou Apple. Jedná se o jediné oficiální vývojové prostředí, které umožňuje překlad, instalaci a ladění aplikací pro iOS. Vzhledem k tomu, že je výsledný software určen téměř výhradně – kromě knihovny EasyPunch – k provozu na zařízeních se systémem iOS, je překlad jinými nástroji než těmi v Xcode zbytečný. Ke zdrojovým souborů jsem nepřiložil Makefile, ale pouze projektové soubory Xcode.

Program mikrokontroléru vývojového kitu Arduino, jsem implementoval přímo v integrovaném vývojovém prostředí dodávaném s kity Arduino. Vývojové prostředí umožňuje překlad programu, verifikaci a následné programování samotného mikrokontroléru.

3.3 Reálné nasazení

Reálné nasazení s provedením uživatelských testů jsem naplánoval a připravil událost OSP – oblastní žebříček Jihomoravské oblasti pořádaný mým domovským klubem Tesla Brno 20. dubna 2013 v Předklášteří u Tišnova. Z důvodů vadné tiskárny na prototypovém terminálu jsem byl nucen toto reálné nasazení posunout na pozdější dobu. Novou tiskárnu jsem ihned objednal u výrobce v USA, ale do dnešního mi ji nedodal. Další pokus o reálné nasazení budu mít pravděpodobně 25. května 2013 na závodech oblastního žebříčku Jihomoravské oblasti v Kuřimi. Vše bude záležet na domluvě s pořadatelem.

Prototypový terminál i celou práci jsem tedy alespoň představil několika organizátorům událostí v OSP. Jejich reakce byly vesměs velmi pozitivní s nabízením případné podpory. Z reakcí také usuzuji, že ač tito organizátoři vidí těžkosti v současném stavu výpočetního zajištění událostí v OSP, nad řešením příliš neuvažovali.

Abych mohl do odevzdávané technické zprávy zahrnout, alespoň nějaké výsledky uživatelských testů, sestavil jsem z oslovených dobrovolníků testovací skupinu a provedl testy na simulovaném prostředí. V testovací skupině bylo celkem 11 jedinců. 10 z nich vždy tvořilo závodníky a závodnický ruch v daném prostředí a 1 organizátora.

Postupně jsme simulovali prostředí prvního startovního koridoru, prostoru cíle a místo prezence. Aby byly atributy testů co nejpodobnější reálnému nasazení, probíhalo testování v přírodních exteriérech a účastníci testu byly v některých případech před měřením vystaveni fyzické námaze odpovídající běžeckého závodu oblastního žebříčku na krátké trati.

3.4 Uživatelské testy

Pro první reálné nasazení terminálu jsem navrhnul několik uživatelských testů. Díky těmto testům by pro mě mělo být jednoduché empiricky změřit, nakolik jsem realizací práce dosáhl cílů stanovených na začátku. Testy se zaměřují na hlavní imperativy přidané hodnoty projektu. Jsou to především uživatelská přívětivost terminálu a obslužné aplikace. Výrazná redukce počtu kusů výpočetní techniky použitých v průběhu události a zrychlení odbavení závodníků.

Všechny testy jsem navrhoval s ohledem na to, aby každý cíl každého z nich byl specifický, jednoduše měřitelný, dosažitelný, relevantní k testované problematice a časově ohraničený.

3.4.1 Test snížení použitého počtu kusů výpočetní techniky

Měřit redukci počtu kusů výpočetní techniky použitých v průběhu události bohužel v testovacím nasazení není jednoduše možné. Každá činnost prováděná terminálem musí být stejně zálohována starým systémem. Do počtu použitých kusů VT tedy zapojím pouze ta zařízení, která se účastní nového toku dat a nikoliv záložní starý systém. Do testu budou započítány jako jednotky celé počítače, tiskárny, kontrolní jednotky SI, ale pro přesnost i papírové dokumenty, které nově nahradilo použití terminálu. Výsledný rozdíl počtu kusů bude vyjadřovat procentuální pokles počtu nutných zařízení.

Tento test jsem provedl v porovnání se stavem na běžném závodu oblastního žebříčku. Při tomto závodu jsou obvykle potřeba 2 počítače, tiskárna, startovní listina a 1 vyčítací kontrolní jednotka SI. Tedy dohromady 5 kusů. Při použití terminálů tento počet kusů klesá na 2. Jeden terminál na startu závodu a další v cíli. **Úspora počtu kusů výpočetní techniky proti současnému stavu je tedy 60%**.

3.4.2 Test rychlosti vyčtení čipu

Cílem testu rychlosti vyčtení čipu je porovnání doby, strávené závodníkem na vyčítacím stanovišti. Doba vyčtení jednoho čipu je měřena od okamžiku přistoupení závodníka ke kontrolní jednotce do okamžiku odebrání vytištěného protokolu závodníkem. V současném systému je doba jednoho vyčtení poměrně dlouhá, protože tisk protokolu nebývá okamžitý. Většinou se na papír formátu A4 tiskne více protokolů naráz a před odebrání se musejí rozřezat. U vyčítacího stanoviště se tak tvoří nechtěné shluky a fronta závodníků.

V testu empiricky změřím průměrnou dobu výčtu 20 závodníků současným systémem a průměrnou dobu výčtu čipu 20 závodníků pomocí prototypu terminálu. Rozdíl průměrných dob vyjádřený v procentech mi napoví, jakého zrychlení je možné díky terminálu dosáhnout.

V testu nebudu uvažovat nestandardní situace, jako jsou reklamace výsledků. Všichni testovaní závodníci přijdou s prototypem terminálu do styku poprvé.

Vestavěná tiskárna terminálu v době provádění simulovaných testů nefungovala. Tento test, jsem tedy nezařadil.

3.4.3 Test zrychlení odbavení závodníka po doběhnutí

Dalším rychlostním testem jež lze zařadit do kategorie měření optimalizace procesů je měření doby od označení cílové jednotky závodníkem po nahrání výsledných dat do organizačního systému. Cílem testu je zjištění časového rozdílu mezi současným systémem a použitím prototypu terminálu.

Výsledná data se do organizačního systému nahrají okamžikem vyčtení dat, tedy akustickou a zvukovou signalizací kontrolní jednotky. Jelikož z výčtu čipu je přesně znán – a do databáze uložen – čas ražení cílové kontrolní jednotky i čas vyčtení, budu test provádět až se statistickými daty

získanými během závodu. Do testovacího vzorku závodníků tedy bude možné zahrnout všechny závodníky, kteří v závodě doběhli.

Rozdíl průměrných dob odbavení závodníka po doběhnutí vyjádřím opět v procentech.

Pro simulaci testu jsem definoval typický setup pro závod, tedy prostor cíle závodu v místě prezence (vzdálenost od cíle k místu prezence 300 m). Testovací vzorek byl 10 jedinců, které jsem rozdělil na 2 poloviny. Obě skupiny běžely stejný orientační závod. V cíli probíhalo odbavení jedné skupiny klasickým způsobem (oražení cílové jednotky, vydechnutí, odevzdání mapy, občerstvení, cesta k místu prezence, přečtení dat z čipu SI) a druhé skupiny pomocí terminálu (oražení cílové jednotky, vydechnutí, přečtení dat z SI čipu, odevzdání mapy, atd.). Průměrná doba odbavení závodníka současným způsobem byla 4 minuty 52 sekund. Průměrná doba odbavení závodníka terminálem byla 48s. **Doba odbavení závodníka v cíli se zkrátila přibližně o 83,57%.** Zajímavým zjištěním v rámci testu byla spokojenost závodníků s časným tiskem protokolu jejich výsledků ještě před odevzdáním mapy.

Vzhledem k poruše zabudované tiskárny jsem k tisku protokolů použil externí jehličkovou tiskárnu účtenek a pro její použití jsem upravil program mikrokontroléru terminálu.

3.4.4 Test zrychlení odbavení závodníka v prvním startovním koridoru

V prvním startovní koridoru závodník musí v současném systému oznámit operátorovi startu číslo svého SI čipu. Operátor startu nalezne závodníka ve startovní listině, kde ho označí a pustí do startovního koridoru. V případě, že operátor zjistí, že závodník startuje v jiné minutě, musí ho buď vykázat z koridoru nebo rovnou připustit ke startu. Existují také situace, kdy je třeba změnit v dokumentu číslo čipu SI či startovní čas.

Cílem testu je změřit změnu průměrné rychlosti odbavení jednoho závodníka. V rámci testu změřím dobu odbavení v prvním startovním koridoru 20 závodníků odbavených současným systémem a 20 závodníků odbavených pomocí prototypu terminálu. Měřit budu dobu od začátku hlášení čísla SI čipu po ukončení odbavení operátorem. Rozdíl průměrných dob odbavení závodníka vyjádřím v procentech.

V simulovaných testech do startovního koridoru postupně přicházeli testovací jedinci. V každé startovní minutě prošel koridorem jiný počet závodníků. Testovací jedinci prošli koridorem vždy vícekrát, tak aby byla dodržena skupina 20 vzorků v obou přístupech. Doba odbavení jednoho závodníka současným systém byla průměrně 12 sekund. Doba odbavení závodníka terminálem byla průměrně 6 sekund. **Doba odbavení se tedy díky terminálu snížila o 50%**.

3.4.5 Dotazník pro uživatele systému

Posledním krokem, který jsem zařadil do uživatelského testování bude dotazník pro uživatele prototypu terminálu. Dotazník povedu formou osobního interview a jeho přesný obsah určím na základě poznatků získaných během prvního nasazení. Rámcově, ale budu především zjišťovat subjektivní hodnocení práce s terminálem. Zdali považují respondenti terminál za zlepšení stavu a v čem vidí jeho největší přínos. Dále potom, kde naopak vidí největší slabiny systému, co by terminálu vytkly a změnily a pokusím se je motivovat k zamyšlení nad dalším rozvojem terminálu.

Tento test jsem zatím neměl šanci uskutečnit. Zařadit jej do simulovaného testování nemělo význam.

4 Závěr

Úkolem této práce bylo navrhnout a realizovat prototyp vstavěného terminálu pro podporu organizace událostí v orientačních sportech. Cílem práce bylo integrovat veškeré prvky výpočetní techniky potřebné během organizace závodů do jednoho přenosného zařízení. Dále pak usnadnit a zrychlit přípravu událostí po technické stránce, zrychlit odbavení závodníků na startu i v cíli a zjednodušit operátorům i závodníkům ovládání jednotlivých prvků výpočetní techniky.

V rámci práce bylo nutné detailně nastudovat pravidla závodů v orientačních sportech, dokumentaci k dominantnímu časoměrnému systému SPORTIdent i v současnosti používané řešení pro podporu organizace. Dále bylo nutné seznámit se s teorií vestavěných a mikroprocesorových systémů a vývojovými kity Arduino. Práce i tak vyžadovala mnohé prerekvizitní znalosti z oblasti programování a vývoje aplikací pro systém iOS.

Realizaci práce je možné rozdělit do 4 částí. První je návrh a vývoj hardwarového prototypu terminálu, který obsahuje mikrokontrolér, miniaturní termální tiskárnu, základovou stanici SPORTIdent, mikropočítač a obal. Druhou částí je návrh protokolu Punch pro přenos časoměrných dat a vývoj multiplatformní knihovny EasyPunch implementující protokol. Třetí částí je návrh a vývoj klientské aplikace terminálu EasyEvent, sloužící jako uživatelský interface terminálu a poskytující potřebnou funkcionalitu pro podporu organizace událostí. Čtvrtou částí je provedení uživatelských testů při zkušebním nasazení.

V rámci vývoje terminálu a především obslužné aplikace EasyEvent jsem se především snažil o vytvoření co nejpoužitelnějšího řešení, které výrazně zkrátí a usnadní organizační procesy. Velkou pozornost jsem proto věnoval definici skupin uživatelů a návrhu uživatelského rozhraní aplikace.

Prototyp se podařilo vytvořit a zprovoznit, ale kvůli problémům s poruchovostí hardwarové komponenty tiskárny bylo testovací nasazení několikrát přesunuto a prozatím nemohly proběhnout uživatelské testy. S nasazením je počítáno v nejbližších dnech. Reálné uživatelské testy jsem nahradil alespoň testy simulovanými. Výsledky ve všech testech ukázaly výrazné zlepšení měřených hodnot při použití terminálu.

V dalším rozvoji práce bych se chtěl věnovat vytvoření serverové části pro podporu organizace větších závodů a postupnému rozšiřování navrženého systému do dalších oblastí organizace.

Literatura

[1] SCHWARZ, Josef, Richard RŮŽIČKA a Josef STRNADEL. *Mikroprocesorové a vestavěné systémy*. Brno, 2006. Studijní opora. VUT Brno, FIT.

[2] Apple Inc.: Core Data Programming Guide [online]. Dostupné z: http://developer.apple.com/library/ios/#documentation/cocoa/Conceptual/CoreData/cdProgrammingGuide.html

[3] HENYCH, Martin. *VÝUKA ORIENTACE NA ZŠ POMOCÍ MAP PRO ORIENTAČNÍ BĚH*. Brno, 2009. Bakalářská práce. Masarykova Univerzita. Vedoucí práce doc. PaedDr. Eduard Hofmann, CSc.

[4] ČESKÝ SVAZ ORIENTAČNÍCH SPORTŮ. *Sborník pravidel orientačních sportů* [online]. 2013 [cit. 2013-05-12]. Dostupné z: http://www.orientacnisporty.cz/cz/dokumenty/

[5] INTERNATIONAL ORIENTEERING FEDERATION. *IT Commission Documents* [online]. 2013 [cit. 2013-05-12]. Dostupné z: http://orienteering.org/it-commission-documents/

[6] FUTURE TECHNOLOGY DEVICES INTERNATIONAL LTD. *FT232R USB UART IC Datasheet* [online]. 2010 [cit. 2013-05-12]. Dostupné z: http://www.ftdichip.com/Support/Documents/DataSheets/ICs/DS_FT232R.pdf

[7] SPORTIDENT GMBH. *SPORTIdent Programming Guide* [online]. 2011 [cit. 2013-05-12]. Dostupné z: http://www.sportident.com/index.php?option=com_kunena&view=topic&catid=5&id=4&Itemid=2280&lang=en

[8] INTERNATIONAL ORIENTEERING FEDETRATION. *IOF Data Standard 2.0.3*. 2010. Dostupné z: http://orienteering.org/resources/it/data-standard-2-0/

[9] INTERNATIONAL ORIENTEERING FEDERATION. *IOF Data Standard* [online]. 2013 [cit. 2013-05-12]. Dostupné z: http://code.google.com/p/iofdatastandard/

[10] ALLAN, Alasdair. *IOS sensor apps with Arduino* [online]. Beijing: O'Reilly [cit. 2013-05-08]. ISBN 978-144-9308-483.

[11] ADAFRUIT. *A2 Micro panel thermal printer* [online]. 2012 [cit. 2013-05-08]. Dostupné z: http://www.adafruit.com/datasheets/A2-user%20manual.pdf

[12] TEXAS INSTRUMENTS. *MAX232 Datasheet: Dual EIA-232 drivers/receivers* [online]. 2004 [cit. 2013-05-08]. Dostupné z: http://www.ti.com/lit/ds/symlink/max232.pdf

[13] Wikipedia: Voltage regulator [online]. Dostupné z: http://en.wikipedia.org/wiki/Voltage_regulator, 2013 [cit. 2013-05-08].

[14] SHIMPI, Anand Lal a Brian KLUG. Apple iPhone 4S: Thoroughly Reviewed: Battery Life. *Anandtech* [online]. Indianapolis, Ind.: Que, 2011, [cit. 2013-05-08]. Dostupné z: http://www.anandtech.com/show/4971/apple-iphone-4s-review-att-verizon/15

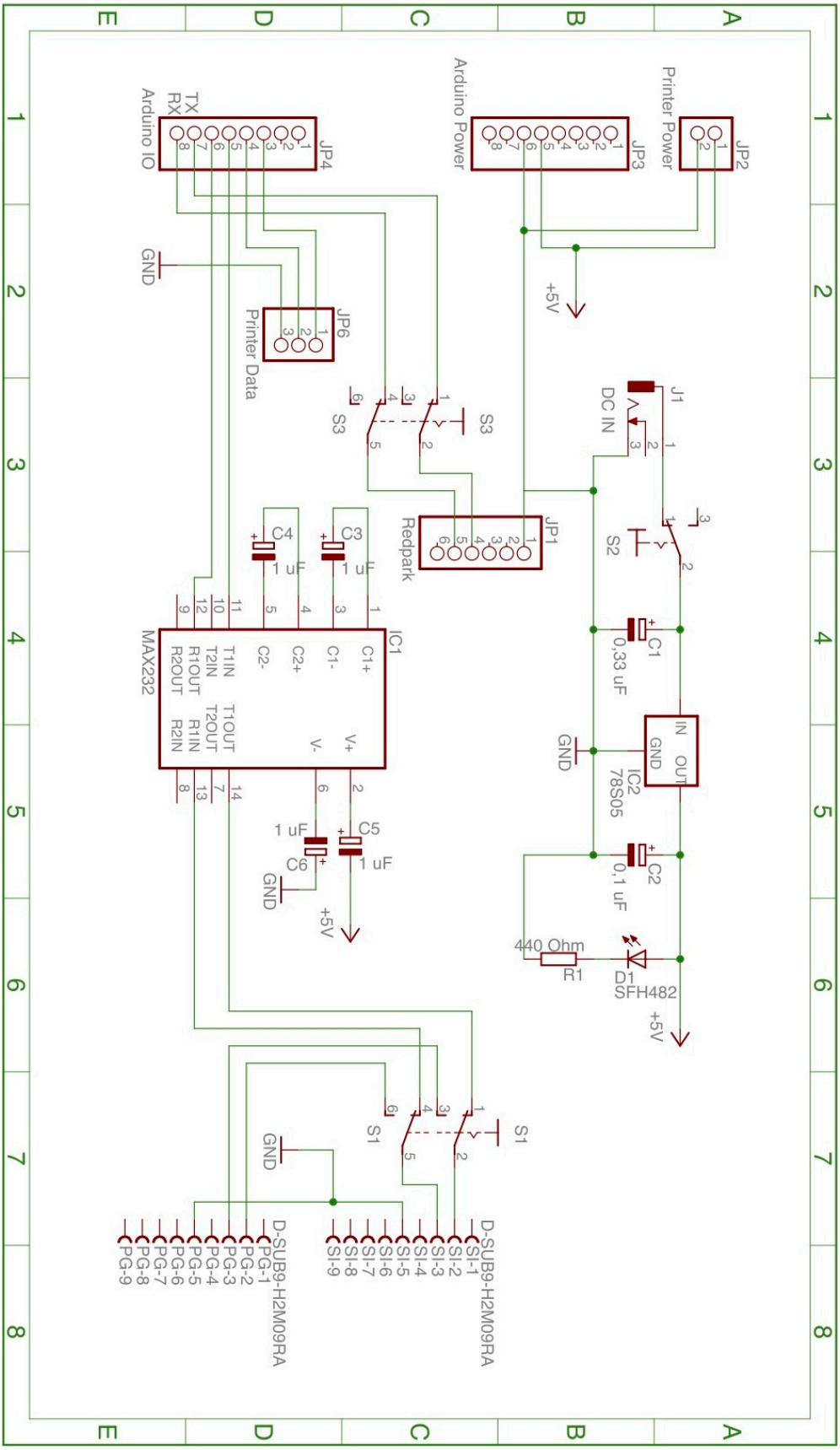
[15] Wikipedia: Watt [online]. Dostupné z: http://en.wikipedia.org/wiki/Watt, 2013 [cit. 2013-05-08].

Seznam příloh

Příloha 1 – Výpočet kontrolního součtu

```
//** Autor: Jurgen Ehms
//** Description: Programm to generate 16 BIT CRC
//** Return values: 16 BIT CRC
//** Version    last change
//** 1.00       07.09.2004
#define POLYNOM 0x8005
unsigned int crc(unsigned int uiCount,unsigned char *pucDat){
    short int iTmp;
    unsigned short int uiTmp,uiTmp1,uiVal;
    unsigned char *pucTmpDat;
    if (uiCount < 2) return(0);        // response value is "0" for none or one byte
    pucTmpDat = pucDat;
    uiTmp1 = *pucTmpDat++;
    uiTmp1 = (uiTmp1<<8) + *pucTmpDat++;
    if (uiCount == 2) return(uiTmp1);   // response value is CRC for two data bytes
    for (iTmp=(int)(uiCount>>1);iTmp>0;iTmp--) {
        if (iTmp>1) {
            uiVal = *pucTmpDat++;
            uiVal= (uiVal<<8) + *pucTmpDat++;
        } else {
            if (uiCount&1) {             // odd number of data bytes, complete with "0"
                uiVal = *pucTmpDat;
                 uiVal= (uiVal<<8);
             } else {
                uiVal=0; //letzte Werte mit 0
             }
        }
        for (uiTmp=0;uiTmp<16;uiTmp++) {
            if (uiTmp1 & 0x8000) {
                uiTmp1  <<= 1;
                if (uiVal & 0x8000)uiTmp1++;
                uiTmp1 ^= POLYNOM;
            } else {
                uiTmp1  <<= 1;
                if (uiVal & 0x8000)uiTmp1++;
            }
            uiVal <<= 1;
        }
    }
    return(uiTmp1);
}
```

Příloha 2 – Schéma zapojení terminálu



Příloha 3 – Třída BSx6 a BSx7

Třída BSx6 a BSx7

Speciální třída určená pro zařízení komunikující pouze v módu BSx6 resp. BSx7. Definice zpráv je totožná s dokumentací výrobce systému SPORTIdent [7].

Třída BSx6e a BSx7e

Třídy zařízení definující veškeré zprávy z tříd BSx6 resp. BSx7, ovšem zasílané v rozšířeném módu. Parametry zpráv jsou shodné. Rozdíl je pouze ve formátu zprávy a v kódu. Kódy zpráv jsou sice také shodné, ale jsou expandované na délku 3 bajty (zpráva ve třídě BSx6 s kódem 0x5D je ve třídě BSx6e s kódem 0x00005D).

Díky těmto třídám je možné přenášet razící data od základových stanic přes různé transportní vrstvy až k místu zpracování.

Příloha 4 – Třída Mini Thermal Printer

Třída Mini Thermal Printer protokolu Punch definuje zprávy pro komunikaci s termální tiskárnou závodních protokolů. Pro komunikaci s tiskárnou existují 4 zprávy klienta a jedna zpráva hostitele. Klientské jsou zpráva pro zapnutí a vypnutí tiskárny, zpráva pro dotaz na stav tiskárny, zpráva pro nastavení parametru tisku a zpráva pro tisk znaků. Hostitelská pak je zpráva o stavu tiskárny.

STX					TRN	CRC (2B)	ETX
0x02	0x00	0x01	0xA0	0x01	0x??	0x????	0x03

Tabulka 1 – Zpráva pro vypnutí/zapnutí tiskárny

Zpráva s kódem 0xA0 slouží klientovi pro aktivaci či deaktivaci tiskárny a obsahuje 1 parametr o velikosti 1 bajt udávající stav vypnuto, tedy 0x00 a zapnuto, tedy 0x01. Kvůli úspoře energie by měl klient tiskárnu při delší nečinnosti vypínat.

STX					CRC (2B)	ETX
0x02	0x00	0x01	0xA1	0x00	0x????	0x03

Tabulka 2 – Zpráva pro dotaz na stav tiskárny

Zpráva s kódem 0xA1 slouží pro dotaz na stav tiskárny a neobsahuje žádný parametr. Pokud je tiskárna připojená, měla by odpovědět zprávou s kódem 0xA2.

STX					STS (2B)	CRC (2B)	ETX
0x02	0x00	0x01	0xA2	0x02	0x??	0x????	0x03

Tabulka 3 – Zpráva o stavu tiskárny

Zpráva s kódem 0xA2 slouží tiskárně pro odeslání svého aktuálního stavu. Parametr STS je rozdělen na jednotlivé bity následovně:

- **1. bit** označuje, je-li tiskárna zapnutá (0 – vypnutá, 1 – zapnutá),
- **2. a 3. bit** označuje nastavenou velikost textu (10 – malá, 01 – střední, 11 – velká),
- **4. bit** signalizuje nastavený tučný text (0 – normální, 1 – tučný)
- **5. bit** signalizuje nastavený podtržený text (0 – nepodtržený, 1 – podtržený),
- **6. bit** signalizuje inverzní podklad textu (0 – černý text na bílém, 1 – bílý text na černém),
- **7. a 8. bit** signalizuje zarovnání (10 – doleva, 11 – na střed, 01 – doprava),
- **9. až 12. bit** uvádí hodnotu rozestupu řádků v 16 různých hodnotách (0000 – nejmenší rozestup, 1111 – největší rozestup).

Ostatní bity těla zprávy jsou rezervované pro budoucí použití.

STX					STS (2B)	CRC (2B)	ETX
0x02	0x00	0x01	0xA3	0x02	0x??	0x????	0x03

Tabulka 4 – Zpráva pro nastavení parametrů tisku

Zpráva s kódem 0xA3 slouží pro nastavení parametrů tisku. Je totožná se zprávou 0xA2, s rozdílem, že touto zprávou není možné tiskárnu zapnout ani vypnout, první bit těla zprávy by měl být vždy nastaven na 1 a tiskárna jej musí ignorovat.

Na zprávu 0xA3 by tiskárna měla nastavit parametry tisku a odpovědět klientovi zprávou 0xA2.

STX					CHARS	CRC (2B)	ETX
0x02	0x00	0x01	0xA4	0x??	0x??	0x????	0x03

Tabulka 5 – Zpráva pro tisk znaků

Zpráva 0xA4 slouží k zaslání znaků tiskárně. Zpráva může být libovolně dlouhá, ale musí správně nastavovat délku těla zprávy. Pokud, je tiskárna zapnuta provede tisk a na zprávu neodpovídá.

Příloha 5 – Obsah paměťové karty

- Plakát
- Video
- Zdrojové kódy aplikace EasyEvent
- Zdrojové kódy knihovny EasyPunch
- Zdrojové kódy programu mikrokontroléru
- Textová zpráva ve formátu docx a pdf